
BERICHTE

AUS DEM FACHBEREICH INFORMATIK
DER UNIVERSITÄT OLDENBURG

HERAUSGEBER: V. CLAUS, P. GORNY, C. MÖBUS

Klassifikationen, Konzepte und Modelle
für den Mensch-Rechner-Dialog

Axel Viereck

1906

DISSERTATION

NUMMER 1/87 - MÄRZ 1987

BERICHTE

**AUS DEM FACHBEREICH INFORMATIK
DER UNIVERSITÄT OLDENBURG**

HERAUSGEBER: V. CLAUS, P. GORNY, C. MÖBUS

**Klassifikationen, Konzepte und Modelle
für den Mensch-Rechner-Dialog**

Axel Viereck

DISSERTATION

NUMMER 1/87 - MÄRZ 1987

Referent: Prof. Dr. P. Gorny
Korreferenten: Prof. Dr. C. Möbus, Prof. Dr. F. Nake
Tag der Disputation: 19.12.1986

Zusammenfassung

Abhängig von den Kenntnissen, Fertigkeiten und Fähigkeiten eines Benutzers fällt dessen Einschätzung über die Benutzerfreundlichkeit bzw. -unfreundlichkeit eines interaktiven Computer-Systems aus. Es ist deshalb nur selten möglich ein bestimmtes Dialogverhalten generell als benutzerfreundlich oder -unfreundlich zu deklarieren, sondern diese Begriffe sind allgemeiner auf einer abstrakten Ebene zu definieren. Losgelöst von einzelnen Anwendungen betrachten wir deshalb im ersten Teil der Arbeit den Mensch-Rechner-Dialog abstrakt als eine Folge von Dialogschritten aus Aktionen und Reaktionen zwischen einem Benutzer und einem Rechner und untersuchen diese Folge nach funktionalen und Konkretisierungs-Gesichtspunkten auf allgemeingültige, immerwiederkehrende Formen und Ausprägungen. Dabei nehmen wir Klassifikationen vor, durch die das formale Rechner-Kommunikationsverhalten beurteilbar wird.

Diese Klassifikationen werden dann im zweiten Teil der Arbeit als systematisches Gerüst zur Beurteilung allgemeiner Richtlinien, Begriffe und Vorgehensweisen herangezogen, um Konzepte zur Gestaltung des Mensch-Rechner-DIALOGS zu entwickeln. Dabei erweist sich die Forderung nach einem anwendungs- und geräteunabhängigen System - dem sogenannten allgemeinen Dialograhmen - als Bindeglied zwischen einem Benutzer und einem Anwendungsprogramm zur Durchführung des Mensch-Rechner-DIALOGS als grundlegend.

Möglichkeiten zur Realisierung des allgemeinen Dialograhmens werden dann im dritten Teil der Arbeit aufgezeigt. Dazu bilden wir mit Hilfe von erweiterten Petri-Netzen - den sogenannten strukturierten, markierten Netzen - eine Reihe von Modellen, die auf der Grundlage der vorher entwickelten Klassifikationen und Konzepte die dynamischen Aspekte, sowie die Abhängigkeiten und Wirkungen von Aktionen und Reaktionen innerhalb des Mensch-Rechner-DIALOGS darstellen. Die Formulierung der Modelle in Pascal-ähnlicher Notation ist dabei realisierungsnah und ein erster Schritt auf dem Weg zur Implementierung eines allgemeinen Dialograhmens.

Anstelle eines Vorworts

Für die Betreuung der Arbeit, die stete Unterstützung und die vielen anregenden Diskussionen, ohne die das Gelingen der Arbeit wohl nicht möglich gewesen wäre, möchte ich mich bei Peter Gorny bedanken. Ebenso gilt mein Dank Claus Möbus und Frieder Nake für ihre Bereitschaft zur Durchsicht und Beurteilung der Arbeit.

Mein ganz besonderer Dank geht aber an Conni, die es trotz ihrer anstrengenden Berufstätigkeit die letzten Jahre über auf sich genommen hat, mehr als nur ihren Teil an der Hausarbeit und der Versorgung unserer drei Kinder zu übernehmen und damit die Voraussetzungen zum Entstehen dieser Arbeit geschaffen hat.

Inhaltsverzeichnis

1. Einleitung	4
Teil 1: Klassifikationen	6
2. Begriffsklärungen	7
2.1. Der Mensch-Rechner-Dialog	7
2.2. Benutzertypen	8
2.3. Benutzerfreundlichkeit	10
3. Klassifikation nach dem Konkretisierungsgrad	13
3.1. Programm-Konzeption	14
3.1.1. Partizipative Systementwicklung	14
3.1.2. Erscheinungsbild des Systems	14
3.2. Abstrakte Dialogstruktur	15
3.3. Konkrete Dialogstruktur	19
3.4. Dialogrealisierung	21
3.4.1. Ausgaberealisierung	21
3.4.2. Eingaberealisierung	23
4. Funktionale Klassifikation	26
4.1. Anwendungsdialog	26
4.2. Kontrolldialog	26
4.3. Gestaltungsdialog	27
4.4. Metadialog	27
Teil 2: Konzepte	30
5. Zum Ist-Zustand von CUU	31
5.1. Drill and Practice Programme und Tutorielle Programme	32
5.2. Neuere kommerziell angebotene Software zum Thema CUU	35

6. Zur Programmkonzeption von CUU-Systemen	39
6.1. Bemerkungen zum Erscheinungsbild von CUU-Systemen	39
6.2. Zum Dialogverhalten von CUU-Systemen	41
6.3. Das Konzept des allgemeinen Dialograhmens	43
6.4. Zur Architektur des allgemeinen Dialograhmens	45
7. Zur abstrakten Dialogstruktur von CUU-Systemen	48
7.1. Zum Dialogverhalten von CUU-Systemen	48
7.2. Abstrakte Dialogstruktur, Dialogtypen und Benutzertypen	49
7.3. Eine abstrakte Dialogstruktur für den allgemeinen Dialograhmen	54
8. Zur konkreten Dialogstruktur von CUU-Systemen	58
8.1. Zum Dialogverhalten von CUU-Systemen	58
8.2. Piktogramme	62
8.3. Cursor und Fenster	65
8.4. Dialogstufen	66
9. Zur Dialogrealisierung von CUU-Systemen	68
9.1. Zum Dialogverhalten von CUU-Systemen	68
9.2. Eine Dialogrealisierung für gelegentliche Benutzer	69
9.3. Eine Dialogrealisierung für geübte Benutzer	75
9.4. Zur Einbettung anwendungsorientierter Funktionen	76
9.5. Zwischenformen bei der Realisierung	78
10. Dialoggrundfunktionen für den allgemeinen Dialograhmen	79
10.1. Geräteorientierte Dialoggrundfunktionen	79
10.2. Benutzerorientierte Dialoggrundfunktionen	80
Teil 3: Modellierung	83
11. Ein Modellierungskonzept für Mensch-Rechner-Dialoge	84
11.1. Petri-Netze	84
11.2. Markierte Petri-Netze	88
11.3. Strukturierte, markierte Netze	91
11.4. Modelle für einen einfachen Dialog	93
11.4.1. Organisationsmodell eines einfachen CUU-Dialogs	93
11.4.2. Systemmodell eines einfachen CUU-Dialogs	95

12. Organisationsmodelle für den allgemeinen Dialograhmen	106
12.1. Ein Grundmodell für den allgemeinen Dialograhmen auf der Organisationsebene	107
12.2. Das Organisationsmodell für den allgemeinen Dialograhmen	108
13. Systemmodelle für den allgemeinen Dialograhmen	113
13.1. Ein erstes Systemmodell mit den Klassen Dialogeröffnung, Dialogbeendigung und normaler Dialog	113
13.2. Ein zweites Systemmodell für den allgemeinen Dialograhmen zur Berücksichtigung von Geräte- und Anwendungs- unabhängigkeit	119
13.3. Ein drittes Systemmodell für den allgemeinen Dialograhmen mit der Dialogform Unterbrechung	126
13.4. Bemerkungen zu den Systemmodellen	137
14. Attributsmarken und Instanzenvorschriften für das Systemmodell des allgemeinen Dialograhmens	138
14.1. Allgemeine Vereinbarungen	138
14.2. Algorithmen und Typen zur Beschreibung der Dialogeröffnung	145
14.3. Algorithmen und Typen zur Beschreibung eines Dialogschritts im normalen Dialog	156
14.4. Algorithmen und Typen zur Beschreibung von Unterbrechungen	172
14.5. Bemerkungen zu den Algorithmen	183
15. Zusammenfassung und Ausblick	185
Literaturverzeichnis	188

1. Einleitung

Durch die Tatsache, daß heute interaktiv mit Computern gearbeitet werden kann und Rechnersysteme dadurch in nahezu alle Bereiche des täglichen Lebens Einzug gehalten haben, rückt die Schnittstelle zwischen Menschen und Computern erneut und unter neuen Gesichtspunkten verstärkt in den Vordergrund.

Lange Zeit beschränkt auf Betrachtungen von Geräteanordnungen, Tastaturformen, Flimmerfreiheit von Bildschirmen usw. (Hardware-Ergonomie), ist heute das Dialogverhalten, also die software-mäßige Gestaltung der Mensch-Maschine-Schnittstelle (Software-Ergonomie), als wichtiges Forschungsgebiet anerkannt. Insbesondere der speziellen Problematik, daß immer mehr DV-ungeübte Benutzer mit Computern arbeiten (müssen), wird dabei große Beachtung geschenkt: Der Begriff der Benutzerfreundlichkeit eines Systems ist heute sowohl bei der Entwicklung, wie beim Verkauf von interaktiven Systemen eine hohe Maxime.

Was aber ist Benutzerfreundlichkeit in Mensch-Rechner-Dialogen? Zwar kann man nach den Untersuchungen von Maaß /Maaß84/ den menschlichen Begriff der Kommunikation auf den Dialog eines Menschen mit einem Computer übertragen, ist man sich über unterschiedliche Eigenschaften von Menschen und damit unterschiedlichen Benutzergruppen bei der Mensch-Rechner-Kommunikation einig, welche Benutzergruppen es gibt, ist allerdings strittig (siehe die Diskussion in Kapitel 2) und wie der Dialog an sich beschaffen ist, d.h. seine speziellen Eigenschaften, sein Aufbau aus Grundelementen, ist nicht ausreichend erforscht. Aus diesem Grund ist es einleuchtend, daß die heutigen Forderungen zur Benutzerfreundlichkeit von Computer-Systemen - bereits manifestiert in Form eines DIN-Entwurfs (vgl. /DIN66234/) - nur unpräzises Allgemeingut enthalten, das für den Entwurf von interaktiven Systemen zwar als Leitlinie geeignet ist, nicht aber eine konkrete Unterstützung im Einzelfall darstellt.

Wir wollen in dieser Arbeit an diesem Punkt ansetzen und zunächst in einem ersten Teil versuchen, den Mensch-Rechner-Dialog systematisch zu untersuchen, für ihn Klassen zu bilden und Grundelemente herauszufinden, sodaß auf dieser Basis die vielfältigen Forderungen zur Benutzerfreundlichkeit geordnet und konkret anwendbar gemacht werden können. Es wird sich zeigen, daß eine Klassifikation auf zwei Betrachtungsebenen sinnvoll ist. Zum einen muß der Dialog nach Konkretisierungs-Gesichtspunkten (Grundkonzept, abstrakter Aufbau aus einzelnen Elementen, den sogenannten abstrakten Dialogformen, deren Konkretisierung und Realisierung) untersucht werden (Kapitel 3), zum anderen sind funktionale Gesichtspunkte für den Mensch-Rechner Dialog zu

berücksichtigen, d.h. zu fragen, welcher Zweck (anwendungsbezogene Funktionen, Kontroll-, Gestaltungs- oder Metafunktionen) hinter einzelnen Dialogschritten steckt (Kapitel 4).

Wie ihre Anwendung, bezogen auf die einzelnen Klassen und Grundelemente des Mensch-Rechner-Dialogs, vorzunehmen ist, wird dann im zweiten Teil der Arbeit untersucht. Hier wählen wir den heute viel diskutierten Bereich des computerunterstützten Unterrichts (CUU) als Ausgangspunkt und stellen zunächst die möglichen Einsatzformen vor (Kapitel 5). Danach nehmen wir das heute vorzufindene schlechte Dialogverhalten von Systemen für CUU zum Anlaß, Konzepte zu formulieren, wie die Benutzerschnittstelle solcher Systeme unter Berücksichtigung der vorgenommenen Klassifizierungen und der allgemeinen Forderungen benutzerfreundlich zu gestalten ist (Kapitel 6 bis Kapitel 10).

Als zentrales Konzept wird sich hierbei die Idee eines geräte- und anwendungs-unabhängigen Systems für die Verwaltung und Organisation des Mensch-Rechner-Dialogs erweisen - heute auch in einigen anderen Arbeiten gefordert (vgl. z.B. /Balzert86/, /Coutaz84/), bisher aber noch nicht realisiert. Mit der Definition solch einer Ebene zwischen einzelnen Anwendungsprogrammen und speziellen Ein- und Ausgabegeräten wird versucht, die Verantwortung für benutzerfreundliches Systemverhalten aus den Händen der Anwendungsprogrammierer zu nehmen und stattdessen zentral zu berücksichtigen.

Wie man ein solches System aufbauen und realisieren kann, ist dann Thema des dritten Teils dieser Arbeit. Mit Hilfe der Klassifikationen des ersten und der Konzepte zur Gestaltung benutzerfreundlicher interaktiver Systeme des zweiten Teils entwickeln wir Modelle zur Darstellung der statischen und dynamischen Aspekte von Mensch-Rechner-Dialogen bei einem geräte- und anwendungs-unabhängigen System. Dazu bedienen wir uns attribuierten, markierten Petri-Netzen, für die wir zunächst die Modellierungsweise vorstellen (Kapitel 11), entwickeln dann ein die Architektur des konzipierten Systems wiedergebendes Netz (Kapitel 12 und Kapitel 13) und geben für die einzelnen Instanzen Algorithmen zur Beschreibung ihrer Funktionsweise und für Kanäle Datenstrukturen zur Darstellung der im System ausgetauschten Nachrichten an (Kapitel 14). Auf diese Weise schaffen wir eine Grundlage, auf der zukünftige Realisierungen aufbauen können.

Teil 1: Klassifikationen

Losgelöst von einzelnen Anwendungen betrachten wir im ersten Teil der Arbeit den Mensch-Rechner-Dialog abstrakt als Folge von Dialogschritten aus Aktionen und Reaktionen zwischen einem Benutzer und einem Rechner und untersuchen diese Folge nach funktionalen und Konkretisierungs-Gesichtspunkten auf allgemeingültige, immerwiederkehrende Formen und Ausprägungen. Wir entwickeln dabei eine Klassifizierung nach dem Konkretisierungsgrad und eine funktionale Klassifikation, die die verschiedenen Ebenen und Zwecke des Mensch-Rechner-DIALOGs deutlich machen und die es erlauben, den bisher so verschwommenen Begriff der Benutzerfreundlichkeit von interaktiven Systemen mit seinen diversen Einzelforderungen systematisch auf Mensch-Rechner-DIALOGe anwenden zu können.

2. Begriffsklärungen

Angesichts der Begriffsvielfalt, fehlender Konventionen und Definitionen und der damit verbundenen unterschiedlichen Interpretationen von Bezeichnungen im Bereich interaktiver Computersysteme wollen wir die systematische Untersuchung des Mensch-Rechner-Dialogs beginnen mit einer Diskussion der drei Begriffe, die für die gesamte Arbeit grundlegend sind: "Mensch-Rechner-Dialog", "Benutzer" (eines interaktiven Computersystems), "Benutzerfreundlichkeit" (eines interaktiven Computersystems).

2.1. Der Mensch-Rechner-Dialog

In einer technischen Sichtweise kann man

Definition Dialog bezeichnen als eine Folge von Schritten (den sogenannten Dialogschritten), jeweils bestehend aus Aktion und Reaktion zweier am Dialog teilnehmender Instanzen (den sogenannten Dialogteilnehmern). Eine Aktion setzt sich dabei zusammen aus dem Senden einer Nachricht durch den einen und dem Empfangen dieser Nachricht durch den anderen Dialogteilnehmer.

Bei dem Mensch-Rechner-Dialog sind die teilnehmenden Instanzen einerseits der Mensch und andererseits der Rechner.

Im Verhältnis zum Dialog zwischen zwei Menschen ist der Mensch-Rechner-Dialog jedoch sehr arm. Ein Rechner verfügt nicht über Mimik und Gestik (nonverbale Kommunikation), ist nicht fröhlich, gelassen oder wütend. Die Prinzipien menschlicher Kommunikation

- die Abhängigkeit eines Dialogs von einem Handlungskontext,
- das Befolgen von Konventionen beim Dialog,
- Zusammenarbeit und
- die Berücksichtigung von Partnerbildern

sind auf den Mensch-Rechner-Dialog nur begrenzt übertragbar. Eine Untersuchung dieser Prinzipien hinsichtlich ihrer Anwendbarkeit auf Mensch-Rechner-Dialoge stammt von Maaß /Maaß84/. Sie resümiert:

"An den Rechner werden (vermittels Programmen und Daten) ein Umweltwissen, Kommunikations-Erwartungen, Intentionen, Kommunikations-Normen, Selbstbilder und Partnerbilder delegiert. Diese normalerweise nicht explizit in Programmen wiederzufindenden, sondern virtuellen Komponenten bestimmen

sein Verhalten in der Interaktion mit Benutzern.

Die Programmierung bedingt ein formales Kommunikationsverhalten des Rechners. Der Benutzer muß sich anpassen, wenn er erfolgreich kommunizieren will." (/Maaß84/, S. 56)

In diesem Sinne ist der Computer für den Menschen somit ein virtueller Dialogpartner mit formalem vom Entwickler des Programms vorgegebenem Kommunikationsverhalten.

2.2. Benutzertypen

Das Einhalten von Normen bei dem Mensch-Rechner-Dialog bestimmt Selbst- und Partnerbild vom Rechner und auch vom Menschen. Dabei muß von unterschiedlichen Bildern für Benutzer, den Benutzertypen, ausgegangen werden.

Im allgemeinen ist bereits die Verwendung des Begriffs **Benutzer** sehr vielschichtig. Er kann als **Nutzer** von Informationstechnik aufgefaßt werden und damit als Mensch oder auch als Institution andere Eigenschaften und Interessen aufweisen als ein Benutzer, interpretiert als **Betroffener** von Informationstechnik (vgl. /Essig81/). Der letztere steht bei der Bearbeitung von Aufgaben meist in direktem Kontakt zum Computer. Dieser zweite Typ wird auch **direkter Benutzer** genannt im Gegensatz zum **indirekten Benutzer**, der - ohne selbst am Gerät zu sitzen - von den Leistungen eines Computers bei seiner Arbeit profitiert (vgl. /Schneider83/). Insofern ist der indirekte Benutzer zwar auch Betroffener - er arbeitet z.B. mit Computerausdrucken -, aber er ist auch schon Nutzer in dem Sinn, daß er Leistungen des eingesetzten technischen Geräts in Anspruch nimmt. Eine andere Bezeichnung für direkte und indirekte Benutzer ist die des **Endbenutzers** oder auch **end user**, auch unterteilt in **direct end user** bzw. **indirect end user** (vgl. /Dehning78/). Wir wollen im folgenden den Begriff Benutzer im Sinne von Betroffener verwenden.

Eine weitere Typisierung anhand der Kenntnisse und Fähigkeiten von Benutzern - hier in der Interpretation von Betroffenen - ist oft zu finden (vgl. /Schneider83/, /Schmitt83/, /Gorny80/) und insbesondere auch für die uns im 2. Teil interessierende CUU-Umgebung wichtig. Danach kann man - die Bezeichnungen differieren von Autor zu Autor - vgl. die Gegenüberstellung von Benutzertypen in /Dehning78/ oder /Schmitt83/ - unterscheiden: der naive Benutzer (naive user), der gelegentliche Benutzer (casual user), der geübte Benutzer (dedicated user). Wir wollen für die nachfolgenden Betrachtungen die drei Benutzertypen wie folgt charakterisieren.

Der **naive Benutzer** weiß nicht, wie ein Computer funktioniert, ein Programm aufgebaut ist; sein Modell von dem Computer ist naiv. Er kann in der Bedienung der Geräte ein Experte sein, wie z.B. die Sekretärin an der Tastatur eines Textsystems oder die Datatypistin an einem Kartenlocher; er kann auf seinem

Fachgebiet ein Experte sein, wie der Ingenieur am CAD-System oder der Physiker bei einer automatisierten Meßwertauswertung; die Vorgänge, wie der Computer ihn bei seiner Arbeit unterstützt, sind ihm unbekannt.

Der **gelegentliche Benutzer** hat im allgemeinen wenig oder keine Erfahrung mit Computern und Dialogsystemen. Möglicherweise kennt er zwar prinzipiell die Abläufe innerhalb einer solchen Informationsverarbeitungsmaschine, bzw. hat diese einmal gelernt, kann eventuell Teile der Kommandosprache eines Dialogsystems, aber in der Bedienung der Geräte und Handhabung von Kommandos ist er ungeübt und unsicher. Unter gelegentlichen Benutzer faßt man z.B. den Benutzer, *"der ein System nur selten benutzt und in der Zwischenzeit die Details (z.B. Namen von Befehlen) vergißt; den Benutzer, der ähnliche Systeme kennt, aber nicht dieses spezifische; und schließlich den Neuling, der Begriffe und Details erst lernen muß"* (Schneider83/, S. 72).

Der **geübte Benutzer** hat Erfahrung im Umgang mit Computern, kennt deren Arbeitsweise und kann die Geräte bedienen. Er ist vertraut mit den Details eines Dialogsystems und auch den Prinzipien, die einer Klasse von Systemen gemeinsam sind, sodaß für ihn der Übergang von einem System zu einem anderen keine großen Probleme beinhaltet.

In Zusammenhang mit dieser Typisierung ist hervorzuheben, daß ein Mensch als Benutzer von interaktiven Systemen nicht starr einem der Typen zugeordnet werden kann, sondern daß einerseits der Mensch im Umgang mit einem System dessen Eigenschaften kennenlernt und dadurch in der Handhabung und Bedienung Fertigkeiten erlangt und so z.B. vom gelegentlichen zum geübten Benutzer aufsteigt (gegenüber diesem System). Bei diesem dynamischen Prozeß werden vom Benutzer bezüglich der Typisierung Zwischenformen zwischen dem geübten und dem gelegentlichen Benutzer angenommen. Andererseits ist ein Mensch gegenüber einem System (zu einem Zeitpunkt) als geübter, gegenüber einem anderen als gelegentlicher Benutzer einzustufen, sodaß auch unter diesem Blickwinkel die Typenzuordnung nicht statisch vorgenommen werden kann. Insofern sind die genannten Typen als "Meilensteine" bei der Benutzung von interaktiven Computersystemen zu verstehen.

Im Bereich CUU sind alle drei Benutzertypen vertreten. Der Schüler ist ein naiver Benutzer, wenn das Unterrichtsmittel für ihn neu ist, er die Funktionsweise nicht kennt und sich auch privat nicht mit der Technologie beschäftigt. Der Schüler kann bereits Informatik-Kurse besucht haben, gelegentlich am Computer gearbeitet haben und ihn nun als neues Unterrichtsmittel benutzen. Er arbeitet in verschiedenen Kursen von Zeit zu Zeit mit dem Gerät an unterschiedlichen Unterrichtsprogrammen. In diesen Fällen ist der Schüler gelegentlicher Benutzer. Wenn der Schüler sich auch privat mit Computern auseinandersetzt, vielleicht selbst einen besitzt und schon einige Erfahrungen im Umgang mit ihm hat, wenn er sich durch Kurse oder Literaturstudium Kenntnisse über dessen Funktions-

weise und über Dialogsysteme angeeignet hat, dann ist er als geübter Benutzer gegenüber CUU-Programmen einzustufen. Ihm werden unterschiedliche Ansätze von CUU-Systemen nicht so viele Schwierigkeiten bereiten wie den vorgenannten Benutzern.

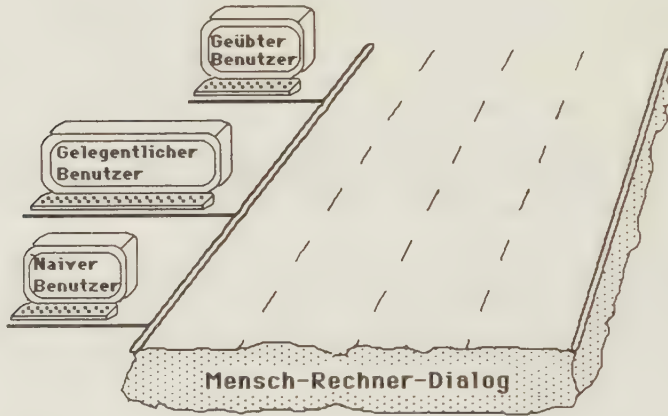


Abb. 2.1: Benutzertypen als "Meilensteine" des Mensch-Rechner-Dialogs

Die Beschreibung der Schülertypen kann man auf die Lehrer als Benutzer von CUU-Programmen übertragen. Auch hier gibt es - vielleicht seltener - den naiven Benutzer, sicherlich häufig den gelegentlichen und den geübten Benutzer. Ein dritte Gruppe im Zusammenhang mit CUU sind die Programmierer, die außer in den Reihen der Lehrer, z.B. in Verlagen oder Softwarefirmen zu finden sind. Geht man davon aus, daß zumindest bei den Lehrern und Verlagen für die Programmentwicklung Autorensysteme vorherrschend verwendet werden, so sind hier fast ausnahmslos gelegentliche und geübte Benutzer anzutreffen. Softwarefirmen beschäftigen durchweg geübte Benutzer.

2.3. Benutzerfreundlichkeit

Abhängig von dem Typ eines Benutzers ist dessen Einschätzung, ob ein System als benutzerfreundlich oder -unfreundlich anzusehen ist. Es wird im Folgenden nur noch zwischen gelegentlichen und geübten Benutzern unterschieden, da der naive Benutzer in Bezug auf sein Verhalten während des Mensch-Maschine-Dialogs sich diesen beiden Typen zuordnen läßt. Was für einen gelegentlichen

Benutzer als angenehm anzusehen ist - z.B. längere Erklärungen, Hilfen - empfindet ein geübter Benutzer als lästig. Er möchte lieber kurze Kommandos, Abkürzungsmöglichkeiten innerhalb des Dialogs, um den Eingabe-Aufwand zu verringern. Es ist deshalb nur selten möglich, eine bestimmte Maßnahme generell als benutzerfreundlich oder -unfreundlich zu deklarieren, vielmehr sind diese Begriffe allgemeiner auf einer abstrakten Ebene zu definieren. Wir wollen eine etwas längere Passage aus einer entsprechenden Definition hier zitieren und danach kurz auf die Normungsbestrebungen zu diesem Bereich eingehen. Ansonsten wird auf die umfassenden Untersuchungen z.B. in /Kühn80/, /Dehning78/ oder /Cakir83/ verwiesen.

"Im Bereich der Informatik und einigen Forschungsansätzen in Großbritannien und Skandinavien wird von einem Harmoniemo­dell ausgegangen. Es besagt, daß die zu Beginn unterschiedlichen Interessen bei richtiger Wahl der Methode zur Unterstützung der Systemgestaltung zu einem gemeinsamen, tragfähigen Konsens führen, der alle beteiligten Interessengruppen zufrieden stellt. Dagegen besagt das Konfliktmodell, daß die unterschiedlichen Interessen aller Beteiligten (Nutzer und Betroffene; Anm. d. Verf.) nur in besonders günstig gelagerten Fällen harmonisieren, i.d.R. wird es zu einem Konflikt kommen, der im Rahmen des Systemgestaltungsprozesses auszutragen ist. (...)

Für Benutzerfreundlichkeit in diesem Sinne (des Harmoniemo­dells; Anm. d. Verf.) gibt es eine Reihe von unterschiedlichen inhaltlichen Ansätzen:

- Benutzerfreundlichkeit durch ergonomische Verbesserungen an den Endgeräten der Mensch-Rechner-Schnittstelle (z.B. Bildschirmarbeitsplätze) konzentriert sich auf physische und geistige Belastungen am Arbeitsplatz.
- Benutzerfreundlichkeit im Sinne einer verbesserten Bedienung von DV-Systemen berücksichtigt neben der Ergonomie weitere Faktoren wie Fähigkeiten, Bedürfnisse, Stellung und emotionale Belastungen des Menschen an der Mensch-Rechner-Schnittstelle.
- Benutzerfreundlichkeit zur Humanisierung des Arbeitslebens befaßt sich neben den Bedienungstätigkeiten mit aufgabenbezogenen Aspekten eines DV-Systems, also auch mit Arbeitsinhalten. Hier gilt es, eine Organisation der Arbeitsabläufe zu finden, die den menschlichen Bedürfnissen nach Selbstverwirklichung, Handlungsspielräumen, Weiterbildung usw. gerecht wird. (...)

Die bisher dargestellten Ansätze beinhalten - nach Ansicht der Vertreter des Konfliktmodells - die Gefahr, als Anpassungsstrategien zur Erhöhung der Akzeptanz eines DV-Systems bei den Betroffenen zweckentfremdet zu werden. Um diesem Trend entgegenzuwirken, wurde daher von ihnen folgender Ansatz zur Herstellung von benutzerfreundlichen Systemen entwickelt:

- Benutzerfreundlichkeit als DV-bezogener Beitrag zur Demokratisierung der Gestaltung des Arbeitslebens ist das Ergebnis eines Prozesses, bei dem alle am Computer-Einsatz Beteiligten frühzeitig und angemessen an Planung, Gestaltung und Implementierung eines DV-Systems mitwirken. Diese sog. partizipative Systementwicklung fördert bei den Betroffenen Lernschritte, die sie langfristig in die Lage versetzen, ihre Interessen zu erkennen und

Maßnahmen zu ihrer Realisierung - entsprechend den anderen Ansätzen - vorzuschlagen. Sie macht damit die anderen Ansätze nicht überflüssig, sondern integriert sie in einem umfassenden Rahmen." (Essig81/, S. 51/52)

Auch wenn diese Bemerkungen zur Benutzerfreundlichkeit sich in erster Linie auf die Bereiche Wirtschaft und Verwaltung beziehen, halten wir die Zweigleisigkeit - hier Harmonie, dort Konflikt - für ein wichtiges Unterscheidungsmerkmal auch für den Bildungsbereich. CUU-Systeme sollen nicht deshalb benutzerfreundlich sein, weil sie dann auf weniger Widerstand stoßen, leichter installiert werden können und z.B. Einsparungen an Lehrern vorgenommen werden können. CUU-Systeme sollen benutzerfreundlich sein, um ihren Zweck, nämlich Unterrichtshilfsmittel zu sein, überhaupt erfüllen zu können. Die Einführung von Computern in Schulen soll keine Euphorie sondern eine kritische Auseinandersetzung mit der Materie auslösen und ebenso Unterrichtssituationen verbessern.

In erster Linie am Harmoniemodell orientieren sich die Bestrebungen zu Normungen zur Benutzerfreundlichkeit des DIN. Hier werden in einem Normentwurf über Grundsätze der Dialoggestaltung allgemeine Kriterien zur Benutzerfreundlichkeit von Mensch-Rechner-Dialogen beschrieben. Im einzelnen werden genannt:

- Der Mensch-Rechner-Dialog muß **problemangemessen** sein, d.h. er soll den Benutzer bei der Lösung seiner eigentlichen Arbeitsaufgabe unterstützen und ihn nicht durch System-Eigenschaften zusätzlich belasten.
- Der Mensch-Rechner-Dialog muß **selbsterklärend** sein, d.h. Einsatzzweck und Einsatzweise muß dem Benutzer erkennbar sein, bzw. ihm auf Wunsch erklärt werden können.
- Der Mensch-Rechner-Dialog muß vom Benutzer **steuerbar** sein.
- Der Mensch-Rechner-Dialog muß **leicht erlernbar** sein.
- Der Mensch-Rechner-Dialog muß **verlässlich** sein, d.h. er muß den mitgebrachten oder durch Arbeit am System gewonnenen Erwartungen entsprechen.
- Der Mensch-Rechner-Dialog muß **fehlertolerant** sein, d.h. das System muß Fehler erkennen, dem Benutzer verständlich machen und über kleinere Fehler "hinwegsehen" können. (vgl. /DIN66234/)

Diese Begriffe sind natürlich interpretierbar und in Beziehung zu den Benutzertypen zu setzen. Andererseits bereitet es große Schwierigkeiten, auf diesem Gebiet operationalisierte Ziele zu formulieren. Insofern sind sie sicherlich eher als Empfehlungen zu verstehen, nach denen interaktive Systeme beurteilt werden können, denn als Normen. In Vorwegnahme der Ergebnisse aus Abschnitt 3.2 wollen wir dieser Aufzählung noch das (ebensowenig operationalisierte) Ziel hinzufügen:

- Innerhalb des Mensch Rechner-Dialogs muß der Mensch entsprechend seiner Fähigkeiten **eigenverantwortlich handeln** können.

3. Klassifikation nach dem Konkretisierungsgrad

Die unterschiedlichen Voraussetzungen beim Menschen und das vorgegebene formale Dialogverhalten des Rechners führen dazu, daß der Mensch sich dem Rechner anpassen muß, damit ein Dialog überhaupt möglich wird. Dies betrifft vor allem drei Ebenen. Zunächst muß der Mensch bereit sein, generell den Rechner als Dialogpartner zu akzeptieren, also auf Rechneraktionen sinnvoll zu reagieren. Zweitens muß er die für den Dialog notwendigen Geräte (Tastatur, Bildschirm, Maus) bedienen und drittens die vom System gegebenen Formalia erlernen und benutzen.

Lange Zeit war diese Anpassung kein Problem, da inhaltliches Interesse an der Anwendung den eigentlichen Dialog in den Hintergrund rückte - einem veränderten Partnerbild wurde auf seiten der Systementwickler durch Schaffung problemorientierter Programmiersprachen Ende der 50er, Anfang der 60er Jahre Rechnung getragen. Mit der zunehmenden Verbreitung von Dialogsystemen in heute nahezu alle gesellschaftlichen Bereiche änderte sich dies: Ergonomie wurde auch bei Computersystemen relevant. Zunächst in Form von Hardware-Ergonomie, ab Ende der 70er Jahre auch als Software-Ergonomie (1978 Gründung des Interessenkreises, später Fachgruppe "Interaktive Systeme" innerhalb der Gesellschaft für Informatik, 1983 erstmalig in der Bundesrepublik eine Software-Ergonomie-Tagung des German Chapter of the ACM).

Heute wird vielerorts über Möglichkeiten zur Anpassung des Dialogverhaltens an Benutzerbedürfnisse diskutiert (vgl. Abschnitt 2.3); es fällt jedoch auf, daß diese Überlegungen meist intuitiver oder heuristischer Natur sind - vgl. z.B. /Schmitt83/ oder /Nievergelt83/ -, denen keine systematische Untersuchung des Mensch-Maschine-Dialogs zugrundeliegt. Erste systematische Überlegungen dazu stammen von Dehning, Essig, Maaß von 1978 /Dehning78/ und von Oberquelle /Oberquelle76/, der zur Beschreibung von Mensch-Maschine-Dialogen Petrinetze verwendet. Beide Ansätze sind allerdings nicht in nennenswertem Umfang weiter verfolgt worden.

Wir wollen zunächst die Ideen von Dehning, Essig, Maaß aufgreifen und weiterentwickeln, um dadurch eine systematische Grundlage zur Diskussion und Beurteilung von Mensch-Maschine-Dialogen zu schaffen. Der Ansatz von Oberquelle wird dann im dritten Teil dieser Arbeit wieder aufgegriffen.

Ausgehend von den Betrachtungsebenen "Abstrakte Dialogmethode", "Konkrete Dialogmethode" und "Technische Realisierung" in /Dehning78/ schaffen wir eine Klassifikation des Mensch-Rechner-Dialogs nach dem Konkretisierungsgrad durch Betrachtung der Ebenen "Programm-Konzeption", "Abstrakte Dialogstruktur", "Konkrete Dialogstruktur" und "Dialogrealisierung".

3.1. Programm-Konzeption

Unter Programm-Konzeption ist der grundlegende Ansatz, die Herangehensweise zu verstehen, wie hinsichtlich Vorgaben und Zielen eine Problematik in ein interaktives Programm umgesetzt wird. Einige Aspekte aus diesem Bereich beeinflussen das Dialogverhalten der Programme und werden im nachfolgenden angesprochen.

3.1.1. Partizipative Systementwicklung

Ein schon im Vorfeld der eigentlichen Programmbenutzung wirkender Aspekt zur Programm-Konzeption ist die Möglichkeit, zukünftige Benutzer auf die Systementwicklung Einfluß nehmen zu lassen (vgl. Abschn. 2.3).

Solch eine partizipative Systementwicklung mit interaktiven Computerprogrammen als Gemeinschaftsprodukt von Benutzer und Software-Entwickler beugt möglicher grundsätzlicher Ablehnung eines Systems durch den Benutzer vor und hilft, spezielle Benutzer Gesichtspunkte bei der Systementwicklung berücksichtigen zu können.

Partizipation betrifft hier einerseits das Einbringen von Benutzerwünschen in die Benutzerschnittstelle eines Systems, d.h. welche Geräte zu verwenden sind, wie Ein- und Ausgaben erscheinen sollen usw. In diesem Rahmen wird Partizipation allseits befürwortet und oft angewendet. Andererseits meint Partizipation aber auch Mitbestimmung von Benutzern in der Hinsicht, daß diese über das "ob überhaupt", "in welcher Organisationsform" und die Konsequenzen von interaktiven Systemen mitentscheiden können sollen. Abgesehen von der dieser Mitsprache oft entgegenstehenden fehlenden technischen und fachlichen Kompetenz von Benutzern, beschneidet diese Form der Partizipation das Direktionsrecht in den Unternehmen und Verwaltungen und stößt deshalb auch auf politischen Widerstand. Allenfalls werden Personal- und Betriebsräte über Entscheidungen informiert.

3.1.2. Erscheinungsbild des Systems

Unter diesem Gesichtspunkt wollen wir Aspekte des Dialogverhaltens zusammenfassen, die psychologische Wirkungen beim Benutzer hervorrufen. Ein interaktives System kann durch sein Dialogverhalten den Eindruck der "Rechner-Vollkommenheit" erwecken,

- indem es z.B. durch eine Ausgabenflut den Benutzer überfordert, verunsichert und bei ihm Hemmungen und Minderwertigkeitskomplexe erzeugt;
- indem es durch "schulmeisterliche Belehrungen" den Benutzer einschüchtert und verängstigt.

Ein interaktives System kann weiter demotivierend und langweilig wirken oder beim Benutzer die Phantasie ansprechen, Neugierde erzeugen, ihn herausfordern und dadurch positiv auf seine Dialogbereitschaft einwirken. Ein sehr bedeutender Aspekt in diesem Zusammenhang ist der der Kontrolle. Ein Benutzer, der das Gefühl hat, ständig durch das System beobachtet, angetrieben und überwacht zu werden, wird stets unbefriedigt und ängstlich am interaktiven System arbeiten. Da Angst Lernen behindert, wird er umständlich und mit vielen Fehlern arbeiten, was nach entsprechenden Belehrungen wiederum zu weiterer Unsicherheit im Umgang mit dem System führt. Interaktive Systeme müssen dieser Tendenz entgegenwirken, das gesellschaftliche Urteil des "Überwachungsapparates" Computer ist umzuwandeln in das eines Kommunikationspartners. In diesem Sinne sind "aktive Hilfesysteme" /Lutze85/ wegen der mit ihnen einhergehenden Kontrolle des Benutzers skeptisch zu beurteilen.

Zuletzt sei hier der Aspekt der Isolation genannt. Dialogsysteme werden meist als Einzelarbeitsplätze konzipiert, oft durch Klimatisierung und Beleuchtung und auch räumlich von anderen Arbeitsplätzen entfernt. Dadurch kann der Benutzer weniger mit anderen Mitarbeitern kommunizieren. Sein Geborgenheitsgefühl innerhalb seiner Gruppe, Abteilung o.ä. schwindet, und die Arbeitsunlust wächst. Weiter werden seine Kommunikationsfähigkeiten sich im Laufe der Zeit denen des interaktiven Systems annähern. Seine viel weiter gehenden menschlichen Kommunikationsfähigkeiten nehmen ab. Hier muß über Alternativen nachgedacht werden, die sicherlich einerseits die Entwickler von interaktiven Systemen betreffen, andererseits aber zu einem großen Teil in die Arbeitsorganisation fallen (lassen wir hier einmal die Verwendung solcher Systeme außerhalb des Arbeitslebens außer acht).

3.2. Abstrakte Dialogstruktur

Die zweite Betrachtungsebene für den Mensch-Rechner-Dialog bezeichnen wir als abstrakte Dialogstruktur. Hierunter ist die Abfolge der in einem System benutzten Formen innerhalb einer Dialogschrittfolge zu verstehen. Bezeichnet man die Kombination Aktion des einen und Reaktion des anderen Teilnehmers als Dialogschritt, so wird jeder Dialog aus einer Folge von Dialogschritten, der Dialogschrittfolge, aufgebaut. Jede einzelne mögliche Form wird **abstrakte Dialogform** genannt. Beispiele sind das Menü oder das Kommando.

Dehning, Essig, Maaß /Dehning78/ unterscheiden in ihren Untersuchungen für Mensch-Maschine-Dialoge sechs mögliche abstrakte Dialogtypen und wählen dabei als Ausgangspunkt, von welchem Dialogteilnehmer die Initiative zu einem Dialogschritt ausgeht. Sie beschreiben die **"systeminitiierten Typen"**

- **einfache Anfrage** zur Dateneingabe, bei der der Benutzer durch eine von der Semantik her festgelegte Eingabe auf eine Systemausgabe reagieren soll;

- **Auswahlangebot**, bei dem der Benutzer aus einer vom System vorgeschlagenen Menge von Eingabealternativen auswählt (Ja/Nein-Frage, Menü);
- **Aufforderung mit Antwortsyntax**, bei der der Benutzer auf eine Anfrage mit einer syntaktisch festgelegten Antwort reagieren muß;
- **Aufforderung zur freien Antwort**, bei der der Benutzer aufgefordert wird, sich durch eine quasi-natürlichsprachliche Eingabe zu einem Sachverhalt zu äußern;

und die "**benutzerinitiierten Typen**"

- **Kommando** zur Auswahl einer Funktion aus den in einem Zustand möglichen unter Einhaltung vorgegebener Syntaxregeln;
- **Quasi-natürlichsprachliche Äußerung**, bei der der Benutzer "ohne" syntaktische Einschränkungen eine Funktion aus den möglichen auswählt.

Cakir /Cakir83/ wählt für seine Unterscheidung von "Dialogformen" ebenfalls die Initiative als Ausgangspunkt und führt benutzer- und systeminitiierte Formen an, die im wesentlichen mit denen von Dehning, Essig, Maaß übereinstimmen. Beide Ansätze versuchen damit Kriterien vorzugeben, durch die Mensch-Rechner-Dialoge auf ihre Benutzerfreundlichkeit - bei der die Initiative des Benutzers eine bedeutende Rolle spielt (vgl. die Ausführungen in Abschnitt 2.3) - hin untersucht werden können.

Zwar lassen sich in CUU-Systemen für die oben genannten Typen Beispiele finden, doch ist die angegebene Klassifizierung aus zwei Gründen nicht sinnvoll. Zum einen ist der Unterschied zwischen den Typen "Einfache Anfrage" (1), "Aufforderung mit Antwortsyntax" (2) und "Aufforderung zur freien Antwort" (3) nicht grundsätzlicher Art. Jeder Typ läßt sich problemlos durch Maßnahmen der konkreten Ebene oder Realisierungsebene durch einen anderen ersetzen, es ändern sich für den Benutzer lediglich das Vokabular und Format - er darf beim Typ 3 quasi-natürlichsprachliche Antworten formulieren, statt wie vorher (Typ 1 und 2) exakt nur die Daten einzugeben - sowie die zu beachtenden Syntaxregeln - strenge Einschränkungen beim zweiten Typ gegenüber weniger strengen beim ersten und leichten beim letzten Typ.

Dieserart Unterschiede sind aber nicht Kriterien für diese Abstraktionsebene, sondern erst während der Konkretisierung (vgl. den nächsten Abschnitt) relevant. Insofern sind die genannten Typen auf dieser Ebene als ein abstrakter Dialogtyp anzusehen. Entsprechendes gilt auch für die Dialogtypen "Kommando" und "quasi-natürlichsprachliche Äußerung".

Zum zweiten ist das der Klassifizierung zugrundegelegte Modell der Initiative naiv. Für jeden Rechner existiert ein begrenzter Funktionsvorrat, der - meist durch Benutzerhandbuch - dem Benutzer vorgegeben wird. Durch Kommandos oder quasi-natürlichsprachliche Äußerungen kann der Benutzer eine dieser Funktionen auswählen, und zwar immer dann, wenn er eingabeberechtigt ist, ihn also das System in irgendeiner Weise dazu auffordert. Bei seiner Auswahl hat er

gegenüber dem Menü einen wesentlich größeren Funktionenvorrat zur Verfügung - der aus diesem Grund technisch nicht in der Weise wie Menüs dargestellt wird - und trägt für die Zulässigkeit seiner Aktion selbst die Verantwortung, agiert also eigenverantwortlich. Die Initiative liegt aber bei dem System. Insofern sind alle oben genannten Dialogtypen systeminitiiert. (Die Eingabeberechtigung kann der Benutzer bei einigen Computer-Systemen durch eine "Unterbrechung" (Interrupt) erzwingen. Danach gilt wieder das oben Gesagte.)

Wir wollen der Unterscheidung von /Dehning78/ und /Cakir83/ folgende Klassifizierung gegenüberstellen, die eine neue tatsächlich benutzerinitiierte abstrakte Dialogform beinhaltet und weiter zwischen den Kriterien **Initiative** und **Eigenverantwortlichkeit** differenziert.

- **Die abstrakte Dialogform "Datenabfrage"**

Bei der Datenabfrage wird der Benutzer vom System aufgefordert, die zur Aufgabenerfüllung benötigten Daten einzugeben. Das System übernimmt dabei die Initiative und kann syntaktische oder semantische Vorgaben setzen und dadurch teilweise die Eigenverantwortlichkeit des Benutzers für zulässige Antworten senken. (Formale Überprüfung der Eingabe und Prüfung von Wertebereichsgrenzen.)

- **Die abstrakte Dialogform "Auswahlangebot"**

Beim Auswahlangebot wird der Benutzer aufgefordert, aus einer vom System vorgeschlagenen Menge von Eingabealternativen das gewünschte Element auszuwählen. Sowohl die Initiative wie auch die Verantwortung für die Zulässigkeit einer ausgewählten Möglichkeit ist hier auf Seiten des Systems, d.h. die Eigenverantwortlichkeit des Benutzers ist hier nur gering. Er hat nicht einmal die Möglichkeit, vom Programmierer nicht vorgesehene Antwort-Alternativen zu ergänzen.

- **Die abstrakte Dialogform "Kommando"**

Bei einem Kommando wird der Benutzer vom System aufgefordert, aus der Menge der definierten Funktionen eine gewünschte auszuwählen. Während die Initiative hier wieder beim System liegt, ist der Benutzer jetzt für die Zulässigkeit der ausgewählten Funktion verantwortlich, agiert also in hohem Maße eigenverantwortlich. Wenn die Eingabe alphanumerisch erfolgt, ist er auch für die Syntax voll verantwortlich.

- **Die abstrakte Dialogform "Unterbrechung"**

Unter Eigenverantwortlichkeit und Initiative des Benutzers kann durch diese Dialogform der Programmablauf unterbrochen, d.h. die gerade aktuelle Ar-

beitsumgebung verlassen werden. Das System übernimmt daraufhin die Initiative und setzt den Dialog mit einer der vorgenannten abstrakten Formen fort.

Ein besonderes Problem bei der Konstruktion von Dialogsystemen entsteht durch den Wunsch des Benutzers, den Dialog nur kurz zu unterbrechen und zu einem späteren Zeitpunkt an der Unterbrechungsstelle fortzufahren. Für die Kennzeichnung der Dialogformen spielt es hier jedoch keine Rolle, ob der Benutzer durch die "Unterbrechung" den bisherigen Dialog "abbricht" oder "unterbricht".

Durch Zusammenfassung mehrerer abstrakter Dialogformen ergibt sich die Dialogstruktur. Hierzu sind grundsätzlich verschiedene Vorgehensweisen denkbar. Zum einen kann eine Folge von Dialogschritten aus gleichartigen Dialogformen zusammengesetzt sein und hintereinander ausgeführt werden. Oft zu findende Beispiele für diese Dialogstruktur sind die "Menükonversation" (z.B. in BTX), der "Kommandodialog" oder das "Form-Filling" bei einer Folge von Datenabfragen. Das "Form-Filling" ist gleichzeitig auch ein Beispiel dafür, daß eine Folge gleicher Dialogformen auch zu einem Schritt zusammengefaßt werden kann. In diesem Fall erscheint auf der Ausgabefläche ein Formular (eine "Maske"), in das der Benutzer alle Eingaben einträgt und diese gemeinsam erst nach vollständiger Ausfüllung dem Programm übergibt.

Neben gleichartigen können auch unterschiedliche Dialogformen in Dialogschrittfolgen zu einer Dialogstruktur zusammengefaßt werden. Dies kann einerseits in der Weise erfolgen, daß sich die Aktion eines Dialogschritts in Form einer Dialogform und die Reaktion in Form einer anderen Dialogform darstellt - ein Beispiel dafür ist der Typ Unterbrechung. Andererseits können Aktion/Reaktion von einer Form sein und der Wechsel der Dialogform findet beim Übergang zum nächsten Dialogschritt statt.

Generell ist zur Kombination verschiedener Dialogformen zu sagen, daß der augenblickliche Zustand sowohl für das System als auch für den Benutzer klar erkennbar sein muß. Dem Benutzer muß kenntlich gemacht werden, ob er gerade eine Datenabfrage beantwortet, ob von ihm eine Auswahl oder ein Kommando erwartet wird; er muß wissen, wann und wie er selbst initiativ werden kann.

Ein weiteres Mittel bei der Kombination von Dialogformen zu einer Dialogstruktur sind Möglichkeiten für den Benutzer in Form von bedingten Eingaben (z.B. die Auswahl einer Menüalternative unter bestimmten Bedingungen, andernfalls wird eine andere Alternative gewählt), von Wiederholungen oder Sequenzen (mehrere Eingaben gleicher Art, verschiedenartige Eingaben in einem Schritt) Strukturen dieser Art, die für den Benutzer ein hohes Maß an Dialogkomfort darstellen und in Kommandosprachen durch "Kommandoprozeduren" angeboten werden, existieren in vielen interaktiven Systemen nicht.

3.3. Konkrete Dialogstruktur

Eine abstrakte Dialogform besitzt unterschiedliche Darstellungsmöglichkeiten, z.B. hinsichtlich zugrundeliegender Regeln für die Syntax und Semantik. In einer bestimmten Weise dargestellt wird aus der abstrakten Dialogform eine **konkrete Dialogform**. Eine Folge solcher konkreter Strukturen heißt dann konkrete Dialogstruktur.

Ausgangspunkt der Betrachtungen von Dehning, Essig, Maaß für diese Ebene sind die drei Gesichtspunkte:

- mit welchen Elementen werden Ein- und Ausgaben (Mitteilungen) formuliert,
- in welcher Form treten Mitteilungen auf und
- wie wird der Inhalt in einer Mitteilung dargestellt?

Hieraus leiten sie die folgenden Kriterien zur Einschätzung von Dialogsystemen ab:

1. Das **Vokabular** von Mitteilungen. Dies können natürlich-sprachliche Worte, Codes aus alphanumerischen Zeichen oder Codes aus Symbolen oder grafischen Zeichen (Piktogrammen) sein.
2. Das **innere Format der Eingabe**. Hierunter sind syntaktische Aspekte der Eingabe zu verstehen, wie grammatische Regeln, Längenbeschränkungen, Festlegungen von Reihenfolgen, Vorschriften zur Verwendung bestimmter Zeichen.
3. Das **äußere Format der Eingabe**. Es bezieht sich auf die Gestaltung der Eingabe z.B. hinsichtlich bestimmter Positionen (auf dem Bildschirm) oder der Aufteilung in Spalten und Zeilen.
4. Die **formale Redundanz der Eingabe**. Hierzu zählen Möglichkeiten, zusätzlich zu den benötigten Informationen weitere Daten wie Füllwörter oder Leerzeichen in eine Eingabe mit aufzunehmen.
5. Das **Format der Ausgabe**. Es legt den formalen Aufbau der Ausgabe fest, der sich darstellt z.B. durch vollständige Sätze, Stichworte, Formeln, grafische Darstellungen, Tabellen.
6. Die **semantischen Eigenschaften der Eingabe**. Es bestehen hier die Möglichkeiten, daß widersprüchliche oder überbestimmte Eingaben vom System toleriert oder zurückgewiesen werden können.
7. Die **Interpretationsvorschrift der Ausgabe**. Sie kann für den Benutzer explizit mitgegeben sein oder fehlen, sodaß z.B. Nachschlagen in Handbüchern nötig werden kann.
8. Die **inhaltliche Ausdrucksform der Ausgabe**. Hierunter ist zu verstehen, daß eine Systemausgabe kurz und knapp oder auch lang und ausführlich dargestellt werden kann. Weiter gehören hierzu Aspekte wie Freundlichkeit, Höflichkeit, Widerspruchsfreiheit von Ausgaben.

Die im letzten Abschnitt vorgestellten abstrakten Dialogformen können durch verschiedene Werte dieser acht Kriterien konkretisiert werden. Da es prinzipiell keine Beschränkungen für die jeweiligen Wertemengen gibt, lassen sich auf

diese Weise beliebig viele konkrete Dialogformen konstruieren und durch Kombination konkreter Formen dann auch beliebig viele konkrete Dialogstrukturen.

Ein Beispiel für verschiedene konkrete Dialogformen auf der Basis der abstrakten Dialogform "Datenabfrage" wurde im letzten Abschnitt genannt und soll hier etwas detailliert werden. Wird als abstrakte Dialogform die Datenabfrage verwendet, um eine Uhrzeit, bestehend aus Sekunden, Minuten und Stunden, für das Programm zu erfragen, so kann die die Zeit anfordernde Systemausgabe vom Format her einmal als Stichwort:

"Uhrzeit?"

oder auch als vollständiger Satz auftreten:

"Bitte Uhrzeit eingeben."

Im zweiten Fall ist die inhaltliche Ausdrucksform der Ausgabe noch zu verbessern, z.B. in der Form

"Würden Sie bitte die Uhrzeit eingeben."

In allen Fällen fehlt die Interpretationsvorschrift der Ausgabe: In welcher Reihenfolge sollen in welchem Format die Werte eingegeben werden? Spielt die Eingabeform der Uhrzeit eventuell keine Rolle? Man kann diese Vorschrift zur Ausgabe hinzufügen und damit gleichzeitig auch syntaktische Einschränkungen für die Eingabe festlegen:

"Uhrzeit? (SS,MM,HH)"

"Bitte Uhrzeit eingeben: Stunden:Minuten:Sekunden."

Im ersten Fall wäre eine "richtige" Eingabe

"15,13,10",

im zweiten

"10:13:15".

Das innere Format der Eingabe entspricht im zweiten Fall sehr viel mehr den bundesrepublikanischen Gewohnheiten. Wird auch eine Eingabe

"10 Uhr, 13 Minuten und 15 Sekunden"

vom System verarbeitet, so ist durch formale Redundanz der Eingabe eine

weitere Anpassung an hiesige Gepflogenheiten erreicht. Eine weitere Möglichkeit, die als natürlichsprachliche Äußerung sich noch mehr Benutzergewohnheiten anpaßt, wäre

"13 Minuten und 15 Sekunden nach 10 Uhr".

Nicht praktikabel im Umgang mit Computersystemen ist dagegen eine im Alltag übliche Antwort

"Ungefähr Viertel nach 10".

Das äußere Format der Eingabe kann zusätzlich durch das System festgelegt werden durch eine Anforderung wie

"Uhrzeit? Stunden:
Minuten:
Sekunden:"

Hier ist die Position auf dem Bildschirm, wo die Daten zu notieren sind, genau vorgegeben. Eine ganz andere Möglichkeit für diese Datenabfrage ist durch verändertes Vokabular denkbar: Darstellung einer Analoguhr mit Stunden-, Minuten- und Sekundenzeiger mit der Aufforderung, die gewünschte Uhrzeit durch Bewegen der Zeiger (z.B. mit einem Potentiometer oder der Maus) einzustellen.

3.4. Dialogrealisierung

Auf dieser Ebene betrachten Dehning, Essig, Maaß /Dehning78/ die Möglichkeiten zur technischen Realisierung konkreter Dialogstrukturen auf Ein- und Ausgabegeräten. Dadurch wird die Art, wie sich das Programm gegenüber dem Dialogpartner verhält, entscheidend mitbestimmt. Dies ist aber nun nicht allein eine Frage der zur Verfügung stehenden Geräte, sondern wesentlich auch ein Software-Problem (logische Ein- und Ausgabefläche, Fenstertechnik, Piktogramme). Betrachten wir getrennt nach Ein- und Ausgabe einige Aspekte im Detail.

3.4.1. Ausgaberealisierung

Standardausgabegerät für interaktive Systeme ist der **Bildschirm**. Als Bestandteil einer Rechner-Anlage ist er mit monochromatischer oder heute auch oft Farbdarstellung sowohl für alphanumerische als für grafische Ausgabe ausgelegt, wird jedoch einfachsten ergonomischen Anforderungen (Schwenkbarkeit, Höhenverstellbarkeit, usw.; vgl. z.B. /Benz81/) oft nicht gerecht

und ist durch eine zu geringe Auflösung für grafische Darstellungen nur begrenzt verwendbar (üblich ist etwa eine Zahl von 300 x 600 Pixels; Gorny /Gorny84/ fordert für Grafik-Ausgaben auf einem 15"-Monitor mindestens 675 x 900 Bildpunkte damit der "Treppenstufeneffekt", "Aliasing" vermieden oder abgeschwächt werden kann). Zusätzliche Ausgabebeugeräte sind der Drucker für Texte und der Plotter für grafische Darstellungen.

Bei den meisten Mikrorechnern ist weiter eine akustische Ausgabe durch **Piepen** möglich. Dieses Mittel wird zur Erregung von Aufmerksamkeit in vielen Programmen eingesetzt. Doch ist hier Vorsicht geboten: Zu viele akustische Signale sind lästig und verwirren mehr, als daß sie helfen. Das Geräusch sollte angenehm sein und vom Benutzer in der Lautstärke reguliert, bzw. ganz abgeschaltet werden können. Insbesondere kann die akustische Ausgabe (und Eingabe) belästigend wirken, wenn mehrere Personen in einem Raum arbeiten.

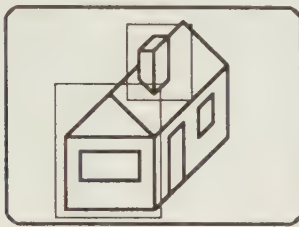
Unterschiedliche Realisierungen konkreter Dialogtypen auf dem Bildschirm entstehen durch Verwendung von Gestaltungsmitteln wie:

- abstrakte Ausgabebläche, virtuelle Ausgabebläche,
- Fenstertechnik (Windowing),
- Auf-, Abrollen des Bildschirmausschnitts (Scrolling), Umblättern (Paging),
- verschiedene Schriftarten, Groß- und Kleinschreibung,
- verschiedene Farben und Helligkeitsabstufungen.
- Blinken und Bewegung.

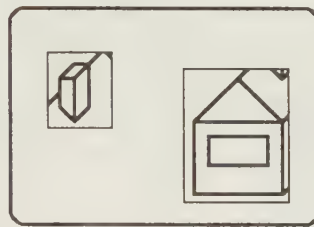
Aus der grafischen Datenverarbeitung bekannt ist die **abstrakte Ausgabebläche** als "künstliche Ebene" zwischen einem Anwenderprogramm und der Bildschirmfläche, auf der zunächst mit Hilfe normalisierter Koordinaten alle Ausgaben durch abstrakte Ausgabefunktionen dargestellt werden. Ein Übertragungsprogramm (Treiber) bildet dann den Inhalt der abstrakten Ausgabebläche oder einen Teil davon auf den Bildschirm ab. Vorteil für den Anwendungsprogrammierer ist die durch diese Vorgehensweise erreichte Geräteunabhängigkeit. (Vgl. auch das "logical device" in /Coutaz85/.)

Mit **virtueller Ausgabebläche** wollen wir eine Ebene bezeichnen, die zwischen dem Anwenderprogramm und der abstrakten Ausgabebläche vorzusehen ist. Ihre Bedeutung ist in Verbindung mit der ebenfalls aus der grafischen Datenverarbeitung bekannten **Fenstertechnik** zu sehen. Das Anwendungsprogramm bildet auf der beliebig großen virtuellen Ausgabebläche alle für ein Problem interessierenden Daten ab, der Bildschirm, bzw. die abstrakte Ausgabebläche beinhaltet immer nur einen gerade relevanten Ausschnitt aus dieser virtuellen Ausgabebläche. Welcher Ausschnitt gerade dargestellt wird, steuert sowohl das Anwenderprogramm wie der Benutzer. Man blickt sozusagen mit dem Bildschirm wie durch ein Fenster auf die virtuelle Ausgabebläche (Vgl. auch das "virtual terminal" in /Coutaz85/.)

Die Vorteile dieser Vorgehensweise sind offensichtlich: Je nach Bedarf können gewünschte Informationen, z.B. allgemeine Erklärungen, Hinweise oder eine Beschreibung von Möglichkeiten - vom Benutzer gesteuert- auf dem Bildschirm dargestellt werden. Insbesondere, wenn man den Bildschirm in unterschiedliche Bereiche aufteilt - in der grafischen Datenverarbeitung **Darstellungsfelder** genannt -, erreicht man eine für den Benutzer hohe Flexibilität bei der Realisierung konkreter Dialogformen. So können Darstellungsfelder für Systemkommandos, Systemmeldungen, grafische Darstellungen, Textdarstellungen usw. vorgesehen werden, in die die entsprechenden Daten aus der virtuellen Ausgabefläche übernommen werden können.



**Virtuelle Ausgabefläche
mit Fenstern**



**Abstrakte Ausgabefläche
mit Darstellungsfeldern**

Abb. 3.1: Fenster und Darstellungsfelder

Spezialfälle der oben beschriebenen Techniken - wenn auch technisch anders realisiert - sind das **Auf- und Abrollen des Bildschirmausschnitts** und das **Umbblättern**. Beide Gestaltungsmittel sind in erster Linie auf Textverarbeitung ausgerichtet und hier schon länger in Gebrauch. Sie erlauben - benutzer-gesteuert - den beliebigen Zugriff auf Textstellen.

Unterschiedliche Betonung von auf dem Bildschirm dargestellter Information erreicht man außer durch örtliche Trennung auch durch Verwendung **unterschiedlicher Schriftarten, Farben, Helligkeitsabstufungen, Blinken und Bewegung**. Zum sinnvollen Einsatz solcher Techniken gibt es innerhalb der Ergonomie Untersuchungen und Erkenntnisse (vgl. z.B. /Nievergelt83/, /Benz81/ oder /Swezey83/).

3.4.2. Eingaberealisierung

Standard-Eingabegerät für interaktive Systeme ist die **Tastatur**. Sie ist in Computer-Systemen in der Regel unterteilt in einen Bereich für alphanumerische und Sonderzeichen, einen für frei programmierbare Funktionstasten, einen für

systembedingte Funktionstasten und mitunter auch einen Bereich mit einer rein numerischen Tastatur. Weitere heute verfügbare Eingabegeräte zur Realisierung konkreter Dialogmethoden sind der **Steuerknüppel**, der **Lichtgriffel**, die **Maus** und neuerdings auch die **Finger** auf einem berührungssensitiven Bildschirm. Tastatur, Steuerknüppel und Maus stehen in Verbindung mit einem Positionsanzeiger auf dem Bildschirm (**Cursor**, **Fadenkreuz**, **Pfeil**), der die Orientierung bei Eingaben ermöglicht.

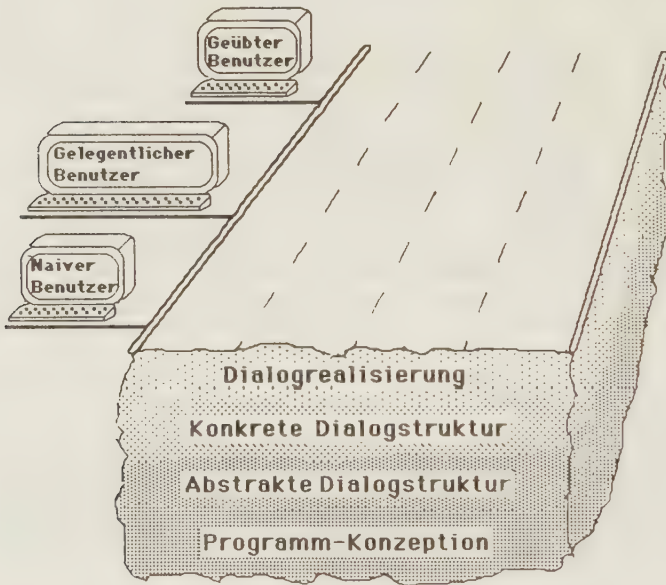


Abb. 3.2: Benutzertypen als Meilensteine und die Klassifikation nach dem Konkretisierungsgrad als Fundament des Mensch-Rechner-Dialogs

Heute für den Benutzer am komfortabelsten angesehen sind die "Eingabegeräte" Lichtgriffel, Maus und Finger, die allerdings nicht als Ersatz sondern als Ergänzung zur Tastatur zu verstehen sind. In Verbindung mit "Fenstern" (s.o.) oder Piktogrammen (vgl. letzter Abschnitt) lassen sich Funktionen leicht ausführen. Veränderungen leicht durchführen und der Programmablauf leicht dirigieren. Es gilt das Prinzip: "*Seeing something and pointing to it is easier for people than remembering a name and typing it*" (/Bewley83/, S. 72). Sind mit dem Lichtgriffel (ständiges Armheben) und dem Finger (ständiges Armheben und

Verschmutzung der berührungssensitiven Bildschirmoberfläche) noch einige ergonomische Probleme verbunden, so ist die Maus heute *"die wohl in Verbindung mit Symbolic-Fenster-Software gelungenste Lösung, hinsichtlich Ergonomie und Komfortabilität"* (/CHIP10.84/, S.17).

Dieses Urteil kann jedoch nur auf die gegenwärtige Situation gerichtet sein, denn mit der Sprach-Ein- und Ausgabe steht heute bereits eine Technik in der Entwicklung, die für Dialogsysteme sehr vielversprechend ist. Insofern sollten allgemeingültige Konzepte hinsichtlich Eingaben für Dialogsysteme nicht auf spezielle Geräte zugeschnitten werden, sondern auf einer höheren Abstraktionsebene basieren und dadurch künftige Neuentwicklungen miteinbeziehen. Dies gilt für eine Konzeption in Bezug auf Ausgabemöglichkeiten ebenso. Ein Mittel, dies zu erreichen, liegt für den Bereich "grafische Datenverarbeitung" vor: Neben geräteunabhängigen grafischen Ausgabefunktionen (abstrakte Ausgabefunktionen, s.o.) werden dort abstrakte grafische Eingabefunktionen definiert (/GKS83/), die durch die zur Verfügung stehenden physikalischen Geräte verwirklicht werden müssen. Diese Vorgehensweise ist allgemein für interaktive Systeme zu übernehmen, um dadurch Dialogrealisierungen allgemeingültig und geräteunabhängig zu erklären.

4. Funktionale Klassifikation

Bisher ist immer allgemein von "dem Mensch-Maschine-Dialog" gesprochen worden. Zwar haben wir im letzten Kapitel unterschiedliche Betrachtungsebenen des Dialogs diskutiert, doch betrafen diese Ebenen in erster Linie die Art des Dialogs. Nicht berücksichtigt sind bisher inhaltliche und zweckbezogene Aspekte des Mensch-Rechner-Dialogs, "worüber und wozu" überhaupt einzelne Dialogschritte geführt werden. Solche Gesichtspunkte legen eine funktionale Klassifizierung nahe. Es wird im folgenden unterschieden: der Anwendungsdialog, der Kontrolldialog, der Gestaltungsdialog, der Metadialog.

Ähnliche Betrachtungen finden sich ebenfalls in /Dehning78/, werden dort allerdings nicht konsequent zu Ende geführt. Der Begriff "Metadialog" wird heute in diesem Zusammenhang oft verwendet, allerdings in der Regel in einer weitergefaßten Interpretation, als wir ihn hier benutzen wollen (vgl. z.B. /Dehning78/ oder /Kupka81/).

4.1. Anwendungsdialog

Das Spektrum möglicher Anwendungsfunktionen für interaktive Systeme ist sehr weit. Es enthält Funktionen für Datenbanken und Informationssysteme, Text- und Grafiksysteme, Tabellenkalkulationsprogramme, Systeme für Lagerhaltung, Buchführung usw. Für den Bereich CUU sind daneben auch Lern- und Simulationssysteme sowie Autorensprachen zu nennen (vgl. Kapitel 5). Weitere Anwendungen sind zukünftig zu erwarten.

Jede solche Funktion dient der Verwirklichung des eigentlichen Ziels bei der Benutzung des jeweiligen Anwendungsprogramms. Ein Anwendungsdialog wird also geführt beim Eingeben von Text innerhalb eines Textsystems, beim Abfragen von Daten eines Informationssystems, zum Setzen von Parametern für eine Simulation oder auch beim Anfertigen eines Computerprogramms. Die dazu verwendeten Funktionen werden - sinngemäß übereinstimmend mit /Dehning78/ oder /Kupka81/ - als **Anwendungsfunktionen** bezeichnet. Nicht zu dieser Gruppe zählen Funktionen, die gewisse Kontroll- und Steuerfunktionen beinhalten, wie z.B. Festlegungen zum Layout eines Textes oder die Auswahl eines Datenbanksystems.

4.2. Kontrolldialog

Bevor der Benutzer mit einem Anwendungsprogramm arbeitet, während oder nach der eigentlichen Anwendung sind Dialogschritte zur Vor- und Nachbe-

reitung nötig. Hierzu gehören u.a. die Dialogeröffnung und -beendigung, Start und Unterbrechung einer Anwendung, Speichern, Ändern, Löschen oder Wiederaufsuchen von Daten, das Ansteuern bestimmter Geräte, das Aufnehmen einer Kommunikation mit beliebigen Instanzen innerhalb eines Rechnernetzes. Durch solche Funktionen steuert der Benutzer den Dialogablauf, legt die zu bearbeitenden Aufgaben fest und bestimmt, wie er ein System benutzen will. Er kontrolliert dadurch den Dialog. Entsprechende Funktionen nennen wir deswegen **Kontrollfunktionen**, die hierdurch entstehenden Dialoge **Kontrolldialoge**.

4.3. Gestaltungsdialog

Wie sich das Programmsystem dem Benutzer darstellt, wird durch konkrete Dialogstrukturen und Dialogrealisierungen festgelegt. Hier geht es um die Form, in der Ausgaben präsentiert werden, Eingaben zu tätigen sind oder der Bildschirm gestaltet ist. Üblicherweise werden diese Eigenschaften vom Systementwickler vorgegeben. Notwendig für benutzerfreundliche Dialoge sind Funktionen, durch die der Benutzer die Gestaltung von Dialogen mitbestimmen kann.

Durch solche sogenannte **Gestaltungsfunktionen** hat er in **Gestaltungsdialogen** Einfluß auf die Art der Präsentation von Daten auf dem Bildschirm (wo welches Fenster erscheinen soll); er kann das Layout von Texten festlegen, die Darstellungsart von Ausgaben bestimmen. Durch Umschaltfunktionen bestimmt er die Ausführlichkeit von Meldungen und die Syntax für Eingaben (z.B. Abkürzungsmöglichkeiten). Weiter kann er Anwendungs- und Kontrollfunktionen auf seine Bedürfnisse hin verändern bzw. neu definieren (ähnlich den Kommando-prozeduren in Betriebssystem-Kommandosprachen). Man spricht heute in Verbindung mit solchen Möglichkeiten auch von einem **adaptierbaren Systemverhalten**: jeder Benutzer paßt das System seinen speziellen Bedürfnissen an.

Geht man aus von der Definition von Dialogen (siehe Kapitel 2), so betreffen Gestaltungsfunktionen ausschließlich die Form ausgetauschter Nachrichten, z.B. in Kontroll- oder Anwendungsdialogen. Beispielsweise besteht innerhalb eines Textsystems der Anwendungsdialog aus Texteingabe und Darstellung des Textes auf dem Bildschirm. Wie der Text eingegeben wird (z.B. Anschlagstärke der Tastatur) und wie der Text dargestellt wird (z.B. Blocksatz mit kleinem Zeilenabstand und einer bestimmten Schriftart und -größe) kann durch Gestaltungsfunktionen festgelegt werden.

4.4. Metadialog

Im Gegensatz zu den Anwendungsfunktionen beziehen sich Metafunktionen immer auf den Inhalt ausgetauschter Nachrichten. Wenn der Benutzer die

Bedeutung von Meldungen nicht kennt, Informationen über den Systemzustand oder Systemvoreinstellungen benötigt oder Probleme mit Parametern von Funktionen hat, muß das System ihm Hilfestellungen geben. Dazu sind Funktionen vorzusehen, die dem Benutzer allgemeine Informationen bieten, indem sie sich indirekt auf die Inhalte zukünftiger Dialogschritte beziehen und dadurch weitere Anwendungs-, Kontroll- und Gestaltungsdialoge ermöglichen. Beispiele solcher **Metafunktionen** sind Benutzeranfragen bzgl.

- der Funktionsweise einer Benutzerstation,
- des Funktionenvorrats eines Systems,
- der Wirkung, des Aufbaus und der Bedeutung von Funktionen und deren Parameter,
- der Systemvoreinstellungen,
- des aktuellen Systemzustandes,
- der aktuell möglichen Funktionen und
- allgemeiner Beschreibungen und Erläuterungen (vgl. /Dehning78/).

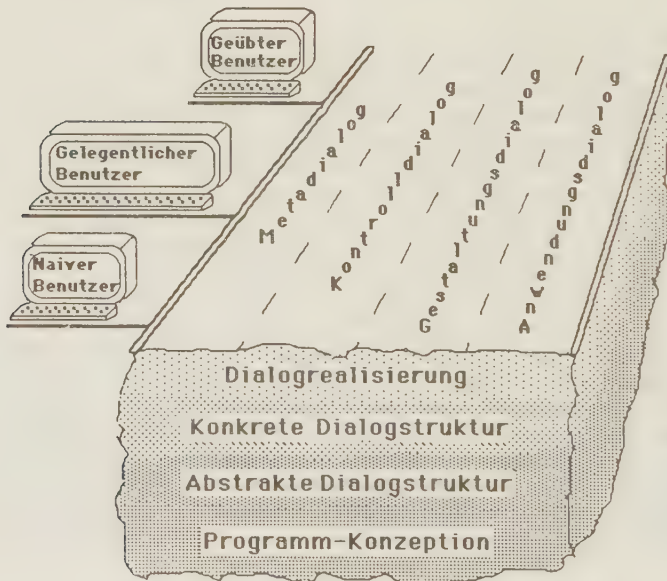


Abb. 4.1: Benutzertypen als Meilensteine, die Klassifikation nach dem Konkretisierungsgrad als Fundament und die funktionale Klassifikation als Fahrbahnen des Mensch-Rechner-Dialogs

Eine Realisierung entsprechender Funktionen ist heute in interaktiven Systemen selten, bereitet aber keine prinzipiellen Schwierigkeiten. Zur Zeit noch nicht realisierbar (oder nur ansatzweise) sind Metafunktionen, die sich auf den Inhalt von Nachrichten bereits zurückliegender Dialogschritte beziehen. Das "Verstehen von Kontext" leisten heutige Computersysteme noch nicht; Fragen bezüglich

- der Bedeutung von Systemausgaben,
- des Sinnes einer Tätigkeit,
- des Grundes für eine Situation

können nur hilfweise durch Zurückführung auf oben genannte Fragen beantwortet werden. Ein solcher Versuch der Antwort sollte durch interaktive Systeme unterstützt werden. Die Entwicklung geeigneterer Verfahren mit Hilfsmitteln der wissensbasierten Systeme bleibt abzuwarten (Auskunfts- & Beratungssysteme, vgl. /Balzert86/).

Teil 2: Konzepte

Im 2. Teil dieser Arbeit wollen wir die in den letzten Kapiteln hergeleiteten Klassifikationen verwenden, um Konzepte zur Gestaltung von Mensch-Rechner-Dialogen zu entwickeln, die - im Sinne des Harmonie-Modells - größtmögliche Benutzerfreundlichkeit gewährleisten. Prinzipiell allgemeingültig auf jeden Mensch-Rechner-Dialog übertragbar, leiten wir diese Konzepte exemplarisch anhand einer Diskussion von interaktiven Systemen zum computer-unterstützten Unterricht her.

Gerade dieser Bereich - der Einsatz von Computern in der Schule und in der Ausbildung - wird heute häufig und kontrovers diskutiert: Welche Bedeutung erlangt der Informatik-Unterricht in der Schule? Brauchen wir eine "Grundbildung Informatik"? Wie soll der Computer in traditionellen Schulfächern eingesetzt werden? Wie sind die Probleme der Lehrerfortbildung zu bewältigen? Welche Art von Computersystemen sind für den Bereich geeignet und notwendig? Wie sollten die einzusetzenden Programmsysteme gestaltet werden?

Wir beschäftigen uns im Folgenden im wesentlichen mit den beiden letzten Fragen und leiten aus den heute verfügbaren und vom Dialogverhalten her fast katastrophal zu nennenden CUU-Programmen allgemeine Anforderungen zur Gestaltung von Mensch-Rechner-Dialogen ab und skizzieren die Architektur von Computer-Systemen für den Bereich des computer-unterstützten Unterrichts, die den Anforderungen gerecht werden können.

5. Zum Ist-Zustand von CUU

Bevor wir uns detailliert mit dem Dialogverhalten von CUU-Systemen auseinandersetzen, wollen wir zunächst den Bereich CUU mit seinen vielfältigen Erscheinungsformen vorstellen, ordnen und einige Begriffe klären.

Kaum ein Bereich innerhalb der Informationstechnologie ist heute von einer Begriffsvielfalt beherrscht, wie die Computernutzung im Bildungswesen. Man findet hier "Computer Aided Education" (CAE), "Computer Aided Learning" (CAL), "Computer-unterstützter Unterricht" (CUU), "Computer-gestützter Unterricht" (CGU), "Rechner-unterstützter Unterricht" (RUU), "Rechner-gestützter Unterricht" (RGU), "Computer-unterstützte Ausbildung" (CUA), "Computer Aided Instruction" (CAI) - Begriffe, die im Großen und Ganzen das Gleiche meinen, nämlich den Einsatz von Computern im Bildungswesen zur Unterstützung von Unterricht. Wir wollen im folgenden unterschiedliche Ausprägungen in der Bedeutung der Begriffe vernachlässigen und, stellvertretend für alle, den Begriff **Computer-unterstützter Unterricht (CUU)** verwenden. In Übereinstimmung mit der Erklärung von Schneider (/Schneider83/) kann CUU charakterisiert werden durch folgende

Definition Computer-unterstützter Unterricht wird - in einem weiten Sinne - verstanden als Unterstützung von Unterricht durch Informationstechnik mit dem Ziel, Lehr- und Lernumgebungen zu verbessern.

Historisch ist der CUU der Idee des "**objektivierten Lernens**" durch "**programmierten Unterricht**" erwachsen. Der Lehrstoff wurde in kleinste Einheiten zerlegt und zunächst in der linearen Abfolge eines Buches bearbeitet. Weiterentwicklungen berücksichtigen dann komplizierte Kontrollstrukturen und Lernkontrollen, die ein lernergerechtes Durcharbeiten und Wiederholen von Lehrinhalten gewährleisten sollten, aber gleichzeitig die Möglichkeiten von Büchern oder Lehrtexten als Medien langsam überschritten.

Ein Ausweg bot sich Anfang der 60er Jahre durch den Computer und die Möglichkeit, ihn als Teilnehmersystem am Sichtgerät im Dialog zu nutzen. Der programmierte Unterricht wurde als Frage-Antwort-Dialog auf Computerprogramme übertragen; die erste Phase des Computers im Unterricht - nach Wedekind /Wedekind81/ der "**klassische CUU**" - hatte begonnen. Inhaltliche Kritik an den Zielen des programmierten Unterrichts, an der mangelhaften Individualisierung, sowie der große Aufwand zur Erstellung geeigneter Lehrprogramme führten in Verbindung mit neuen technischen Möglichkeiten des Computers ab ca. 1970 zu grundlegenden Veränderungen beim CUU. In der zweiten Phase von CUU, die Wedekind "**Rechneradäquate**

Einsatzformen" nennt, kommen Computer-Simulationen, Dialog- und Informationssysteme und interaktive Programmiertechniken für Problemlösen mit Hilfe des Computers zu den vorherigen Einsatzformen hinzu. Der CUU hat sich dadurch losgelöst vom reinen programmierten Unterricht und kann heute vielseitig eingesetzt werden. Daß er sich in der Praxis bisher nicht durchgesetzt hat, hat viele Ursachen. Spezielle Gesichtspunkte zum Dialogverhalten werden in den nachfolgenden Kapiteln ausführlich diskutiert.

5.1. Drill- and Practice-Programme und Tutorielle Programme

Viele Unterrichtssituationen aller Fächer erfordern Einübung von Fertigkeiten, Fakten, Regeln und Methoden des betreffenden Fachgebiets zur Festigung vorhandener Begriffe und Kenntnisse. Als Beispiele seien das Üben von Geschichtszahlen, Vokabeltraining im Sprachunterricht oder das Lösen von Gleichungen zweiten Grades im Mathematikunterricht genannt. Dieses Einüben kann mit Hilfe von sogenannten **"Drill- and Practice-Programmen"** computerunterstützt erfolgen. Dabei stellt das Programm einem Lernenden am Sichtgerät Fragen zu einem Fachgebiet, wertet Antworten aus und kann nach vorgegebenen Kriterien Übungsphasen beenden, wiederholen oder Teile intensivieren.

Fragen werden zufällig oder auch sequentiell aus einer Aufgabensammlung ausgewählt, für Antworten gibt es die Möglichkeit der gebundenen Form (Multiple-Choice, Zuordnung, Umordnung) oder - seltener, weil erheblich schwieriger durch ein Programm zu analysieren - der freien Form.

Geht es nicht um das Verstärken bereits vorher erworbener Kenntnisse und Fertigkeiten sondern darum, neues Wissen und neue Fertigkeiten computerunterstützt zu vermitteln, so spricht man von **"Tutoriellen Programmen"**. Hier wird der Lehrstoff dem Lernenden am Sichtgerät unter Berücksichtigung von Lernerfolg und Lernverhalten präsentiert. Die Steuerung des Lehrablaufs, d.h. welcher Stoff in welcher Form angeboten wird, kann im einen Extrem durch das Programm erfolgen - der Lernende reagiert nur auf Vorgaben des Systems -, im anderen Extrem liegt sie auf Seiten des Lernenden - er kann aktiv sein Tutorium mitgestalten. In der Regel sind Mischformen mit Übergewicht beim ersten Extrem vorherrschend.

Die einzelnen Lerneinheiten sind mit Lernkontrollen und Testfragen durchsetzt und meist durch Übungssequenzen zur Festigung des Stoffes abgerundet. Antworten des Lernenden können - wie oben - in gebundener oder in freier Form vorgesehen werden.

Sowohl Drill- and Practice- wie auch Tutorielle Programme sind als direkte Umsetzung der Methode des programmierten Unterrichts auf das Medium

Computer anzusehen. Die Ziele und Möglichkeiten dieses klassischen CUU unterscheiden sich denn auch nur geringfügig von denen der programmierten Unterweisung (vgl. z.B. die Thesen zu CUU in /Keil78/). Das heutige Angebot entsprechender Programme ist vielfältig. Es stammt meist von Verlagen, Software-Häusern oder Rechner-Herstellern und ist in der Regel zugeschnitten auf spezielle Mikrocomputer. Dazu drei Beispiele aus Werbebroschüren:

"Brüche in Bildern

Klassen: 3-5

1 Diskette DM 88.--

Übungsprogramme mit graphischen Darstellungen. Einfache und gemischte Brüche werden behandelt. Sehr gute Erfolge bei hartnäckigen Fällen von Verständnislosigkeit gegenüber Brüchen.

2 Programme." (/INTUS84/, S.1)

"Englisch/Deutsch- Wörterabfragen

Klassen: Sekundar, Gymnasium

1 Diskette DM 140.--

Englisch/Deutsch oder umgekehrt werden Wörter abgefragt. 800 Wörter sind für folgende 8 Gebiete gespeichert: Easy nouns, easy verbs, easy misc. words, hard nouns, hard verbs and misc. words, harder nouns, harder verbs and misc. words, phrases " (/INTUS84/, S.10)

"Lernprogramm Proportionalität

Im Dialog stattfindendes computerunterstütztes Lernen.

In interaktiver Programmbearbeitung (Zeitbedarf ca. 45 min) wird der Schüler mit dem mathematischen Begriff Proportionalität vertraut gemacht. Er lernt die formale Darstellung (Gleichung) wie die graphische (Ursprungsgerade). Bei Lernschwierigkeiten gibt der Rechner abgestufte Hilfen, alternative Wege werden eingeschlagen. Die Schülerantworten kann der Lernende über Tastatur (Beschreibung erfolgt im Programm) wie über die Paddles eingeben. Schrift und Bilder in hochauflösender Graphik (daher Groß- und Kleinschreibung). Abschluß: 10-Fragen-Test (multiple Choice) mit Bewertung" (/Klett83/, S. 25)

Solche Programme kann ein Lehrender bei den Entwicklern kaufen - er erhält dazu neben der Diskette, die das Programm enthält, oft auch ein Handbuch zur Beschreibung und Benutzung - und auf den in seiner Schule existierenden Rechnern einsetzen.

Daneben ist es aber ebenso möglich, daß der Lehrende Unterrichts-Programme selbst erstellt. Dies kann bei entsprechender Qualifikation und großem Engagement ebenfalls auf einem eigenen oder in der Schule verfügbaren Mikrorechner geschehen. Für Lehrer, die nicht zugleich "Computer-Experten" sind, wurden zu diesem Zweck Hilfen in Form sogenannter **Autorensysteme** oder **Autorensprachen** entwickelt. Beispiele sind die Systeme PILOT von Apple, PLATO von Control Data, COURSEWRITER von IBM, LIDIA von Siemens

und PLANIT von System Development Corp. Einige der Systeme sind (auch) auf Mikrorechnern lauffähig (PILOT, Micro-PLATO). Diese Autorensprachen gliedern sich meist in Teilsysteme zur Erstellung von CUU-Programmen durch Lehrer oder Ausbilder und in solche zur Bearbeitung entsprechender Programme durch Lernende. Für die Programmerstellung wird den Lehrenden eine Art anwendungsorientierter Kommandosprache bereitgestellt; Programmierkenntnisse sind nicht vorausgesetzt. Auch zu den Autorensprachen zwei Beispiel aus Werbebroschüren, bzw. Systembeschreibungen:

"Mit dem Drill&Practice Modell (DPM) können Autoren auch ohne Programmierkenntnisse Testlektionen für das computerunterstützte Training erstellen. Text, verschiedene Frage- und Antwortmöglichkeiten, sequentielle Fragefolgen, Fragen aus einer 'Batterie' nach dem Zufallsprinzip, Feedback und qualifizierte Auswertung sind die wichtigsten Merkmale dieses Lehrmodells. ... Durch die vorprogrammierte Abfolge der einzelnen Lektionsteile sowie die im Modell enthaltenen Präsentationsabfolgen (sequentiell/random) und qualifizierten Bewertungsmöglichkeiten werden bei der Erstellung von Drill&Practice-Lektionen erhebliche Erstellungskosten eingespart."
(/Control Data83/, Drill&Practice-Modell)

"PLANIT ist

- dialogfähig

Ein Benutzer kann Lernprogramme im Dialog erstellen und durcharbeiten. ...

- interpretativ

Der Autor eines Lernprogramms kann sich während der Lernprogramm-erstellung in die Rolle eines Adressaten versetzen, um an jeder beliebigen Stelle der Lernprogrammentwicklung prüfen und revidieren zu können. Änderungen sind leicht möglich und werden sofort in das Programm eingefügt. Verstöße gegen die Syntax von Planit und gewisse semantische Fehler werden bei der Eingabe sofort erkannt und dem Autor mitgeteilt.

- adaptiv

Lerndaten werden automatisch gespeichert. Sie sind jederzeit abrufbar und können als Entscheidungsgrundlage für Programmverzweigungen benutzt werden.

- maschinenunabhängig

Auf allen Rechenanlagen ab einer bestimmten Größe kann Planit leicht implementiert werden da es in der höheren Programmiersprache FORTRAN geschrieben ist. Mit Planit erstellte Lernprogramme sind unter den verschiedenen Benutzern austauschbar.

- leicht erlernbar

Die Unterrichtssprache Planit ist leicht erlernbar, da sie dem Autor eine gut gegliederte Struktur zur Verfügung stellt; die eine flexible und effiziente Lernprogrammierung erlaubt." (Telefunken73/, S. 3)

5.2. Neuere kommerziell angebotene Software zum Thema CUU

Nachdem der klassische CUU als nicht erreichbar und nicht anstrebbare erkannt war, wandte man sich Zielvorstellungen zu, die den Rechner als Hilfsmittel, Werkzeug in den Unterricht integrieren sollten. In Kombination mit verschiedenen Unterrichtsformen kann die "Universal-Maschine" Computer in vielerlei Weise eingesetzt werden.

In Form eines **Taschenrechners** benutzt heute jeder Schüler (etwa ab der 5. Klasse) diese neue Informationstechnik, um kompliziertere Rechnungen durchzuführen. Dieser wird oft, wie früher Logarithmentafel und Rechenschieber, in mehreren Klassensätzen von der Schule bereitgestellt und auch bei Klassenarbeiten und Prüfungen als Rechenhilfsmittel zugelassen. Dabei beschränkt sich sein Einsatz nicht allein auf den Mathematik-Unterricht. Auch in anderen Fächern, wie Physik, Biologie, Chemie und Sozialkunde, können Probleme mit hohem Rechenaufwand praxisnah behandelt werden, ist eine Konzentration durch das Hilfsmittel Taschenrechner auf fachspezifische Inhalte erleichtert. Weiter fördert sein Einsatz die Aufnahme neuer Lehrinhalte in den Unterricht, z.B. quantitative Methoden zur Differential- und Integralrechnung oder statistische Verfahren im Sozialkundeunterricht.

Außer auf den Taschenrechner sei zur Durchführung von Berechnungen hier auch auf klassische CUU-Programme hingewiesen. So bietet z.B. PLANIT einen "Rechner-Modus" an, der es einem Lernenden während der Bearbeitung z.B. eines tutoriellen Programms gestattet, notwendige Berechnungen mit Hilfe des Rechners durchzuführen. Es steht zu erwarten, daß der nicht programmierbare Taschenrechner zukünftig durch **programmierbare Taschenrechner** und **Mikrocomputer** ersetzt wird. Hierdurch wird eine noch bessere Unterstützung bei quantitativen Methoden erreicht, können für den Mathematik-Unterricht Formeln gespeichert werden, ist die Berechnung von Funktionswerten z.B. für die Mathematik noch einmal vereinfacht. (Vgl. hierzu die in /Zimmermann84/ genannten Integrationsmöglichkeiten für Computer im Mathematikunterricht.)

Neben dieser Funktion eines Tischrechners und einer Formel- und Funktionstafelsammlung bietet der Mikrorechner die Möglichkeit, Funktionen und Zusammenhänge grafisch darzustellen. Derartige **Visualisierungen** erleichtern das Verständnis komplizierter Sachverhalte wesentlich und werden in allen Schuldisziplinen als Ergänzung zu oder anstelle von verbalen Beschreibungen oder Tabellen eingesetzt.

Unter Verwendung von Mikrocomputern wird der Einsatz grafischer Darstellungen effektiver; zusätzliche Anwendungen über das hinaus, was Landkarten oder Buchabbildungen zu leisten vermögen, sind möglich. Als Beispiele seien genannt:

- Gegenüberstellung unterschiedlicher zweidimensionaler Projektionen eines 3-D-Objekts,
- Darstellung beliebiger Ausschnitte unterschiedlichen Maßstabs im Verhältnis zum Ganzen,
- Verwendung bewegter Bilder zur Veranschaulichung komplexer Strukturen, Funktionen, Zusammenhänge oder Abläufe,
- Veränderung der Darstellungsart von Objekten (z.B. Farbe, Schattierung, Strichart)

Grafische Datenverarbeitung kann passiv oder interaktiv erfolgen. Wenn sich der Einsatz von Computern zur Visualisierung darauf beschränkt, vorbereitete Programme, die grafische Darstellungen erzeugen, ablaufen zu lassen, spricht man von passiver grafischer Datenverarbeitung. Einwirkungsmöglichkeiten bestehen hier nur durch die eventuell mögliche Vorgabe von Anfangswerten. Fast alle heutigen Anwendungen der grafischen Datenverarbeitung in CUU-Programmen sind von diesem Typ.

Um den Einsatz von interaktiver grafischer Datenverarbeitung handelt es sich, wenn der Lernende aktiv an der Grafikerstellung mitwirkt, indem er über grafische Eingabegeräte wie Lichtgriffel, Maus oder Joystick und über die alphanumerische Tastatur Entscheidungen, Befehle oder Zahlenwerte zu Programmsteuerung eingibt. Ein tutorielles Programm mit Einbeziehung von interaktiver grafischer Datenverarbeitung zur Spannungs- und Strommessung im Physik-Unterricht beschreibt Wagner /Wagner82/.

Heute dient grafische Datenverarbeitung fast ausschließlich der Illustrierung anderer CUU-Formen. Als eigenständiges Konzept zur Unterrichtsunterstützung tritt sie in den derzeitigen Überlegungen nicht auf. Dabei zeigen kommerzielle Systeme wie Lisa und Macintosh (Apple), Star (XEROX), Explorer (Tektronix), CADMUS-WPS (PCS) u.a., welche Möglichkeiten - vor allem auch unter dem Gesichtspunkt der Benutzerfreundlichkeit - sich für die Computernutzung ergeben, wenn man konsequent Techniken der (interaktiven) grafischen Datenverarbeitung zugrundelegt. Wir kommen auf diesen Aspekt in Bezug auf CUU später ausführlich zu sprechen (Kapitel 8 und 9).

Ähnlich zukunftssträchtig wie die grafische Datenverarbeitung ist für den CUU die **Abfrage von Informationssystemen** einzuschätzen. Vor dem Hintergrund einer ständig wachsenden Informationsflut und der abnehmenden Bedeutung von Faktenwissen stellen Informationssysteme ein praktikables Mittel dar, jeweils relevante Daten bereitzuhalten. Der Mikrorechner wird dadurch zu einer Art "intelligentem Lexikon", in dem Lernende zur Beantwortung von Fragen und zur Lösung von Problemen "nachschiessen" können und sich so die für ein Fachgebiet benötigten Kenntnisse selbst erarbeiten. Dies gilt gleichermaßen für allgemeine zentrale Informationssysteme wie für eigene, selbst aufgebaute z.B.

zur Unterstützung von Projekten einer Gruppe. Die zukünftige Entwicklung von wissensbasierten Systemen ("Expertensysteme") wird auch den Zugang zu Wissen erlauben, das erst durch Schlußfolgerungen und Kombination aufzufinden ist.

Heute groß propagiert und nicht nur für den CUU als nahezu universell verwendbar eingestuft werden **Simulationsprogramme**. Ohne näher auf die unterschiedlichen Arten und Vorgehensweisen bei der Simulation einzugehen - vgl. dazu z.B. /Wedekind81/ - wollen wir Simulationsprogramme charakterisieren als durch Computer verarbeitbare, vereinfachende mathematische bzw. formal-logische Darstellungen (Modelle) einer betrachteten Realität. Simulation dient einem zielgerichteten Arbeiten dort, wo entsprechende Arbeiten in der Realität nicht möglich, zu kostspielig, zu gefährlich oder zu zeitraubend wären. Beispiele sind Kalkulationsprogramme, Wirtschaftsmodelle, Planspiele im wirtschaftlichen Bereich, Verhaltensmodelle im gesellschaftlichen Bereich, Simulation von ökologischen Modellen, biologischen, chemischen oder physikalischen Vorgängen im naturwissenschaftlich-technischen Bereich. Diese Vielfalt von Simulationsmöglichkeiten und -verfahren im "täglichen Leben" findet innerhalb des CUU durch eine Vielzahl von Simulationsprogrammen ihr Analogon.

"GALTONA Simulations- bzw. Anwenderprogramm

Dieses Programm simuliert das Galton-Brett auf dem Bildschirm des Apple. Es demonstriert anschaulich die Entstehung der Gauß'schen Normalverteilung. Gezeigt wird eine Kugel (), die durch das Nagelbrett fällt, und jedesmal, wenn sie einen Nagel (^) trifft nach rechts oder links abgelenkt wird. Am Ende des Brettes wird die Kugel aufgefangen und erhöht einen Zähler für die bereits gefallenen Kugeln um Eins." (/Klett83/, S. 23)*

"BALLISTISCHER WURF Simulations- bzw. Anwenderprogramm

Als Eingabe benötigt der Computer die Anfangsgeschwindigkeit, in Betrag und Richtung, sowie sämtliche für den Luftwiderstand eines Körpers relevanten Daten. Nach Berechnung einer geeigneten Dimensionierung des Bildausschnitts zeichnet der Computer in HGR-Grafik zunächst die luftwiderstandslose Flugparabel zum Vergleich und anschließend die reale ballistische Flugbahn." (/Klett83/, S.34)

"RADIOAKTIVER ZERFALL Simulations- bzw. Anwenderprogramm

Das Programm berechnet im ersten Teil Mutter-Tochter- Generationen von radioaktiven Substanzen (auch Gemischen) nach den dafür gültigen Differentialgleichungen und stellt die Ergebnisse grafisch dar; im zweiten Teil simuliert es den radioaktiven Zerfall jedes Teilchens nach seiner mittleren Lebensdauer (random-Simulation), wobei die normale wie die halblogarithmische Darstellung wählbar sind. Die grafischen Darstellungen

können über den grafikfähigen Epson MX-80/82 ausgedruckt werden."
(/Klett83/, S.42)

Diese drei Beispiele sollen andeuten, daß in nahezu allen Fachgebieten heute der Unterricht durch Computersimulationen unterstützt werden kann. Diese sind dann als Ergänzung bzw. Ersatz für Experimente im Chemie-, Physik- und Biologieunterricht anzusehen und können die entsprechenden Fachlehrer bei zeitraubenden Versuchsvorbereitungen entlasten.

Simulationen - im Gegensatz zu Experimenten - "klappen" immer, Sachverhalte können in sehr viel mehr Varianten "durchgespielt" werden, was in der Regel dem Verständnis entgegenkommt. Weiter sei erwähnt, daß in Schulfächern wie z.B. Wirtschaft Inhalte vermittelt werden können, die ohne Simulation nicht denkbar sind (Planspiele, Unternehmensstrategien, Verhaltensmodelle) und daß im naturwissenschaftlichen Bereich auch solche Inhalte "quasi-experimentell" begleitbar sind, die in der tatsächlichen Durchführung zu aufwendig oder zu gefährlich wären.

Ebenfalls zu den neueren Formen von CUU zu zählen ist das **Problemlösen mithilfe von Informationstechnik**. In Informatik-Kursen vorwiegend der Sekundarstufe II lernen Schüler, vorgegebene Probleme zu analysieren und zu strukturieren, einen das Problem lösenden Algorithmus zu formulieren und in ein Computer-Programm umzuwandeln. Als Programmiersprachen finden BASIC, Pascal, LOGO, ELAN und auch MODULA-2 Verwendung oder auch regelbasierte, objektorientierte Sprachen wie PROLOG.

Dieser Bereich von CUU ist heute der am weitesten fortgeschrittene: Lehrpläne legen Unterrichtsinhalte fest, Lehrer - meist solche für Mathematik - bilden sich für den Informatik-Unterricht fort, Schulen beschaffen in wachsendem Maße Mikrorechner. Die Vorgehensweisen sind dabei allerdings sehr uneinheitlich, Konzepte über die zu verwendenden Programmiersprachen und Rechner-systeme gibt es länderübergreifend bisher nicht. Ebenso wird die Lehrerfortbildung uneinheitlich und in unzureichender Weise durchgeführt .

6. Zur Programmkonzeption von CUU-Systemen

Nach den umfangreichen Vorbemerkungen zur Erklärung wollen wir nun zunächst den grundlegenden Ansatz von CUU und seine Konsequenzen für das Dialogverhalten diskutieren und anschließend einige Forderungen zur Programm-Konzeption von CUU-Systemen erläutern.

6.1. Bemerkungen zum Erscheinungsbild von CUU-Systemen

Bestimmend für die Konzeption von Programmen aus dem Bereich "Klassischer CUU" sind die aus der programmierten Unterweisung übernommenen Lernziele und Lernformen. So vielversprechend manchem die Ziele Objektivierung und Individualisierung von Unterricht, Auffangen von Lehrerausfall durch CUU, laufende Information und Kontrolle von Lernstand und -verhalten der Schüler auf den ersten Blick erscheinen mögen, sie spiegeln doch nur den (falschen) Ansatz wider, daß menschliches Denken ausschließlich Informationsverarbeitung und nach den für die "Denkmaschine" Computer geltenden Regeln zu behandeln ist und daß der Wissenserwerb vom sozialen und affektiven Lernen getrennt erfolgen kann.

Es gibt jedoch im Menschen nicht nur reines logisches Denken mit zwei Wahrheitswerten, sondern jedermann denkt auch auf der Grundlage von Gefühlen, Erfahrungen, gesellschaftlicher Stellung (vgl. zur Begründung die Ausführungen zur künstlichen Intelligenz von Bamme /Bamme83/, S. 35ff). Der Mensch ist mehr als nur eine "besonders gute Maschine", der "maschinistische Ansatz" vom klassischen CUU mußte also zwangsläufig scheitern.

Bei den neueren Formen des CUU ist bei Simulationen der maschinistische Ansatz ebenfalls zu bemerken: Ein Modell einer interessierenden Realität verhält sich wie eine Maschine, die richtig gefüttert auch perfekt funktioniert. Stimmt das Ergebnis nicht mit der Wirklichkeit überein, wird die Maschine umgebaut, d.h. das Modell verändert. *"Mit Hilfe von Computersimulationen kann die Wirklichkeit so lange in Modellen nachgebildet werden, bis diese Modelle der Wirklichkeit aufgedrückt werden können."* (/Hubert83/, S. 263) Man schafft sich sozusagen eine künstliche Realität in Form einer Maschine, oft ohne zu berücksichtigen, daß die tatsächliche Realität eben keine Maschine ist und nicht perfekt funktioniert. Das Zurechtfinden innerhalb solcher Scheinwelten kann - bei aller Nützlichkeit von Simulationen - nicht bedeuten, daß man auch in der realen Welt bestehen kann, daß man diese vollständig erkennt. Insofern sollte Computersimulation als Hilfsmittel im Unterricht kritisch betrachtet werden als eine nützliche Form, reale Sachverhalte nachzubilden und zu verdeutlichen, und nicht als "Non-Plus-Ultra" überbewertet werden. Die Wirklichkeit ist durch sie nicht ersetzbar.

Ausgangspunkt für jede Simulation ist immer ein Modell. *"Der Computer ist also ein Spielplatz, auf dem jedes erdenkliche Spiel möglich ist. Man kann Welten erschaffen, in denen keine Schwerkraft existiert oder in denen sich zwei Körper zwar anziehen, aber nicht gemäß dem Newton'schen Gravitationsgesetz (umgekehrt proportional dem Quadrat ihrer Entfernung), sondern nach einem Gesetz der umgekehrten dritten (oder n ten) Potenzen oder Welten, in denen die Zeit in Tanzschritten vorwärts und rückwärts abläuft, entsprechend einer einfachen oder komplizierten Choreographie, wie immer man will."* (Weizenbaum78, S. 157) Das Modell kann - wie Weizenbaum anführt - der Phantasie von Menschen entspringen. Negativ Beispiele dafür sind die zahllosen Computerspiele mit gefräßigen Ungeheuern, Bomben, Raketen usw., oder es kann den Versuch darstellen, eine interessierende Realität nachzubilden. Diese Wirklichkeit wird nun natürlich nicht in ihrer ganzen Komplexität wiedergegeben. Entsprechend des mit der Simulation verbundenen Zwecks wird die Realität vereinfacht, nur dasjenige wird betrachtet, was zum Erreichen des Zwecks wesentlich ist. Dies ist ein sehr problematischer Aspekt, denn in der Auswahl des Wesentlichen liegt - bei sozial- oder wirtschaftswissenschaftlichen Modellen meist mehr als bei naturwissenschaftlichen - schon eine Bewertung der zu untersuchenden Realität.

Dieser Umstand muß dem Lernenden bewußt sein. Dazu ist ihm das zugrundeliegende Modell transparent zu machen und nicht als "Black-Box" vorzugeben; Alternativen sind aufzuzeigen. (Es wäre sicherlich interessant, wie heute gängige "Killerspiele" aussähen, wenn in ihr Modell moralische Gesichtspunkte aufgenommen würden.)

Als letzten Punkt zu diesem Abschnitt wollen wir noch einen grundlegenden psychologischen Aspekt zum Dialogverhalten von CUU-Systemen anführen: die **Isolation**. CUU Programme werden i.a. für Einzelarbeitsplätze konzipiert, d.h. eine Thematik wird von einem Lerner bearbeitet. Diese Arbeitsweise schränkt den Kontakt des Benutzers zu anderen Menschen ein. Zwar kann er zu jedem Zeitpunkt vom Bildschirm aufsehen und die Nachbarn ansprechen, muß dazu allerdings subjektive Hemmungen überwinden, wenn die Nachbarn ebenfalls CUU Programme benutzen, doch ist zu überlegen, ob durch Änderungen am Konzept - z.B. hinsichtlich der Dauer von Dialogen - solche "Kommunikationspausen" vorzusehen sind. Auf keinen Fall darf beim Benutzer der Eindruck entstehen, er werde kontrolliert durch das Programm und dürfe keine längeren Pausen einlegen. Zum anderen stellt sich die Frage, ob solche Kontakte nicht auch über das Lehrsystem anzubieten sind, was natürlich nicht heißen soll, daß das "Schwätzen" mit dem Nachbarn zukünftig computergestützt erfolgen soll. Das Verbinden einzelner Arbeitsplätze ermöglicht aber Zugriff auf gemeinsame Datenbestände, Gruppenarbeit usw. Entsprechende Lehrformen können heutige CUU-Systeme nicht unterstützen.

6.2. Zum Dialogverhalten von CUU-Systemen

Für das Dialogverhalten resultierte aus dem oben Genannten das fast ausnahmslos anzutreffende **statische Konzept für den Ablauf** von CUU- Programmen: Von der durch die Autoren beim Programmmentwurf festgelegten Reihenfolge von Dialogschritten kann man als Benutzer nicht abweichen. Man hat keine Wahlmöglichkeit der Art: 'Dieses Bild nochmals zeigen', 'Langsamer', 'Im Augenblick möchte ich diese Erklärungen noch nicht'; als letzte Rettung bleibt nur der "Aus-Schalter", um den vorgegebenen Ablauf zu verlassen. Oft gliedert sich der zu vermittelnde Lehrstoff in verschiedene Kapitel. Dann besteht für den Benutzer in der Regel die Möglichkeit, über ein Menü das zu bearbeitende Kapitel auszuwählen. Innerhalb eines Kapitels gilt dann wieder das oben Gesagte. Die dieser Konzeption zugrundeliegende simple Programmiermethode mit WRITE- und READ-Anweisungen ist zurückzuführen auf die - wie Böcker sich ausdrückt - **naive Theorie des Lehrens und Lernens**: *"Ich sage dir, was ich weiß, und du merkst dir das; dann frage ich dich, ob du es noch weißt, und wenn du es nicht mehr weißt, sage ich es dir noch einmal."* (Böcker82/, S. 31).

Beim Übergang zur zweiten Phase der Nutzung von Informationstechnologie im Unterricht wurde dieses statische Konzept für den Programmablauf fast nahtlos übernommen. Nur selten hat der Benutzer von z.B. Simulationsprogrammen die Möglichkeit Abläufe abzubrechen, Teile zu überspringen oder zu wiederholen. Eine einmal in Gang gebrachte Simulation wird bis zum Ende durchgeführt und kann erst dann auf Wunsch z.B. mit anderen Werten wiederholt werden. Ein Abweichen von diesem starren Ablauf - z.B. in der oben beschriebenen Form - ist in der Regel nicht vorgesehen.

Ein weiterer Kritikpunkt an der Konzeption von CUU-Programmen ist die fehlende oder nur dürftige **Selbsterklärungsfähigkeit**. Fast ausnahmslos wird ein in gewissen Grenzen erfahrener Computernutzer als Lerner vorausgesetzt, der den Rechner und das Programm als Werkzeug akzeptiert und benutzen will. Selbsterklärungsfähigkeit ist zum einen unter dem Aspekt zu sehen, wie der Rechner und das Programm zu bedienen sind - im wesentlichen betrifft dies die Ebenen "konkrete Dialogmethode" und "Dialogrealisierung" -; wir wollen zum zweiten hierunter auch grundlegende psychologische Aspekte wie **Motivation, Beseitigung von Hemmungen und Ängsten** zusammenfassen. Ein Benutzer ist zum Beispiel im Umgang mit dem CUU-System mit höherer Motivation und mit größerem Selbstbewußtsein bei der Arbeit,

- wenn tutorielle Programme ihre Lernziele, die Vorgehensweise und auch ihre Grenzen zunächst erläutern;
- wenn Simulationsprogramme eine Beschreibung des zugrundegelegten Modells mit Aufzeigen von Vor- und Nachteilen liefern;
- wenn die für Simulationen nötigen Anfangsparameter detailliert erklärt und Beispielwerte gegeben werden.

All dies gibt es zwar hier und da, oft fehlen aber solche Erklärungen, oder sie sind unsystematisch und unvollständig.

Statt durch das Programm den Werkzeugcharakter hervorzuheben, wird mit Hilfe von Eigenschaften des Computers wie Rechengeschwindigkeit und -genauigkeit durch das Programm der Eindruck der Vollkommenheit geweckt. Der Benutzer versinkt in einer Flut von Programm-Ausgaben und reagiert verunsichert. Er ist meist nicht in der Lage, die von Nievergelt formulierten vier Fragen jederzeit zu beantworten, weil das System ihm diese Hilfe nicht zur Verfügung stellt: "*Wo bin ich?*" (*Der Bildschirm sieht nicht wie erwartet aus*); "*Was kann ich hier tun?*" (*Die Menge der aktiven Befehle ist unklar*); "*Wie kam ich hierhin?*" (*Vielleicht wurde eine falsche Taste gedrückt*); "*Wo kann ich hin und wie komme ich dorthin?*" (*Die Möglichkeiten des Systems sollen erforscht werden*)."¹ (/Nievergelt83/, S. 33). Wie wenig vollkommen CUU-Programme für gewöhnlich aber sind, zeigt das von Böcker gegebene Beispiel (C: Computer, L: Lerner):

C: Wieviel ist 2+2 ?

L: vier

C: Falsch. Möchtest Du eine Hilfe haben? Antworte mit 'Ja' oder 'Nein'

L: 4

C: Ich verstehe '4' nicht

L: Wenn Du 4 nicht verstehst, wie soll ich dann von Dir Mathematik lernen können?

C: Richtig" (/Böcker82/, S.31)

Durch Aussagen wie "Ich verstehe '4' nicht" wird versucht, über Schwächen des Programms hinwegzutäuschen. Der Rechner wird durch die Ich-Form personalisiert; es entsteht der Eindruck, der Rechner könne etwas "verstehen". Gerade das kann er aber (zum Glück) nicht. Er arbeitet ein eingegebenes Programm ab, und wenn der Programmautor Konstellationen nicht vorgesehen hat, dann werden bei Auftreten eben solcher Konstellationen entsprechend merkwürdige Situationen entstehen. Diese Unvollkommenheit des Werkzeugs und zugleich die Vorzüge des Menschen mit den Fähigkeiten "Verstehen" und "Interpretieren" dürfen nicht durch WRITE-Anweisungen verschleiert werden, sondern sind dem Lernenden durch das Programm bewußt zu machen.

Inwiefern dieses Hilfsmittel zukünftig mit "Intelligenz" auszustatten ist, um z.B. bisher menschliche Fähigkeiten wie Fehleranalysen oder Bewertungen mit zu übernehmen, wird heute unter dem Stichwort "Intelligenter Computerunterstützter Unterricht" (ICUU) diskutiert (vgl. z.B. /Böcker82/). Hier ergeben sich Möglichkeiten für den Computereinsatz, die weit über das Bisherige hinausgehen. Zu bedenken ist dazu, daß "Intelligenz" von CUU hier nicht mit der Zielsetzung eines Lehrersatzes voranzutreiben ist - dies würde an die frühen Ziele der programmierten Unterweisung anknüpfen und ist sicherlich nicht unproblematisch -, son-

dern im Sinne von Lehrerunterstützung und verbesserter Schülerförderung zu verstehen ist.

Außer Zweifel steht daneben auch die Notwendigkeit von "Intelligenz" von CUU-Systemen hinsichtlich eines anderen Aspekts. (Wenn man in diesem Zusammenhang von Intelligenz sprechen kann.) Ein Dialog ist nicht vollständig charakterisiert durch die ausgetauschten Sätze, Wörter, Daten, sondern es existiert immer ein Gesprächsthema; der menschliche Dialogteilnehmer hat Einstellungen, Kenntnisse und Vorurteile, die einen erheblichen Einfluß auf den Dialog haben. Dieser Punkt findet im allgemeinen bei den Programmautoren zu wenig Berücksichtigung. Ein Dialog über den Dialog - ein **Metadialog** (vgl. hierzu die Klassifikation von Dialog in Abschnitt 3.2.) - kann oft nicht geführt werden und wenn, dann nur in rudimentärer Form, z.B. als "Hilfe-Funktion".

In diesem Bereich ist ein Mehr an "Intelligenz" für CUU-Systeme erforderlich. Die Systeme müssen in der Lage sein, durch ihr Dialogverhalten sich dem Menschen bestmöglich anzupassen und z.B. Fragen der Art: "Wie ist das zu interpretieren?", "Das verstehe ich nicht!", "Hilfe, ich weiß nicht weiter!" zu verarbeiten und sinnvoll darauf zu reagieren. Eine Personifizierung des Rechners sollte aber auch in diesem Zusammenhang vermieden werden. CUU-Systeme müssen weiter "lernende Systeme" sein, die sich in ihrem Verhalten an die sich verändernden Fähigkeiten von Benutzern anpassen können.

6.3. Das Konzept des allgemeinen Dialograhmens

Ausgangspunkt für die Programmkonzeption von CUU-Systemen muß der Ansatz sein, daß durch solche Systeme Hilfsmittel und Werkzeuge geschaffen werden sollen, die Lernumgebungen verbessern. Neben einer Übertragung bestehender Lernformen auf das Medium Computer, sowie der Ergänzung, d.h. Verbesserung dieser Lernformen durch dieses Werkzeug sind dazu verstärkt neue Lernformen auf der Grundlage von CUU zu entwickeln, bzw. weiterzuführen. Hierzu zählen insbesondere auch neuere Formen des CUU, wie Problemlösen mithilfe der Informationstechnik, logische Spiele und Benutzung von Informationssystemen (vgl. Abschnitt 5.2).

Solchen Lernformen werden heutige CUU-Systeme nicht gerecht. Sie beschränken sich - wie gesehen, in denkbar schlechter Art - auf Ergänzung oder Übernahme existierender Lernformen. Einzig durch Möglichkeiten zur Simulation sind Verbesserungen von Lernumgebungen gegeben. Die Programm-Konzeption von CUU-Systemen muß aus diesen Gründen verändert werden. Ein grundlegender Wandel in der Interpretation der Systeme ist notwendig, um die Nachteile, die durch Einzelsysteme, schlechte Systemkonzepte und Benutzerschnittstellen überwiegen, zu vermeiden.

Dazu ist es notwendig - wie bei der Diskussion von Benutzerfreundlichkeit angesprochen (vgl. Abschnitt 2.3) -, daß das Werkzeug als Hilfsmittel von allen Beteiligten akzeptiert wird, daß die Beteiligten mitentscheiden können, ob und wie sie es einsetzen wollen. Grundlegend im Vorfeld der eigentlichen Systeme ist somit eine detaillierte Betrachtung der Technik, eine Aufklärung ihrer Möglichkeiten und Grenzen, die Entwicklung einer Didaktik der Informationstechnologie. Zum letzteren zählt sowohl eine Methodik, wie die Technik im Unterricht üblicher Fächer eingesetzt werden kann, als auch eine Didaktik der Informatik im engeren Sinne, bezogen auf den Informatik-Unterricht an allgemeinbildenden Schulen.

Im Sinne des Harmoniemodells (vgl. Abschnitt 2.3) müssen CUU-Systeme ein Höchstmaß an Benutzerfreundlichkeit bieten. Es ist dazu erforderlich, in der System-Konzeption globale Vorgaben zu entwickeln,

- durch die grundlegende Eigenschaften wie Flexibilität, Transparenz, leichte Erlernbarkeit usw. einerseits für die verschiedenen Benutzertypen und andererseits für die verschiedenen Ziele der Benutzung (Erstellen eines Programms durch Lehrer, Anwenden eines Programms durch Lerner) erreichbar sind;
- durch die größtmögliche Geräteunabhängigkeit erzielt wird, wobei auch zukünftige technische Entwicklungen - soweit heute erkennbar - mit einbezogen sein müssen;
- durch die neben bisherigen CUU-Formen - von denen zumindest teilweise (Drill and Practice, Tutorielle Systeme) zu erwarten ist, daß sie an Bedeutung verlieren - auch neuere unter einheitlicher Arbeitsweise berücksichtigt werden können.

Möglich wird dies durch das Konzept eines **allgemeinen Dialograhmens**. Aufbauend auf einer weitreichenden Systemarchitektur zur Gewährleistung des letztgenannten Aspekts muß dieser Rahmen eine Reihe von Dialoggrundfunktionen zur Verfügung stellen, die einerseits Unabhängigkeit von möglichen Ein- und Ausgabegeräten gewährleisten und andererseits die Möglichkeit bieten, neueste Erkenntnisse der Software-Ergonomie bei der Entwicklung von CUU-Programmen berücksichtigen zu können.

Ein solcher allgemeiner Dialograhmen wird dadurch für den Bereich CUU vergleichbar dem grafischen Kernsystem GKS für die grafische Datenverarbeitung: In Bezug auf die in einer Anwendung zur Verfügung stehenden Geräte wird auf der Grundlage einer universellen Architektur **Geräteunabhängigkeit** erreicht. Entsprechende Gerätetreiber sind an den Dialograhmen anzuschließen. Hinsichtlich Benutzerfreundlichkeit und speziellen Anwendungen gewährleistet dieses Konzept größtmögliche **Anwendungs-unabhängigkeit**, indem einzelne Anwendungsprogramme in den allgemeinen Dialograhmen integriert werden. Unterschiedliche Anwendungsprogramme treten dadurch dem Benutzer in einheitlicher Arbeitsweise gegenüber. Die

Trennung der Benutzerschnittstelle von einer Anwendung wird heute in verschiedenen Arbeiten gefordert (vgl. den "Dialog-" und "Service-Monitor" in /Stork85/, den "dialog socket" in /Coutaz85/, den "User - Agent" in /Bullinger83/ oder die "MCS" in /Balzert86/).



Abb. 6.1.: Funktion eines Allgemeinen Dialograhmens

6.4. Zur Architektur des allgemeinen Dialograhmens

Die Systemarchitektur heutiger CUU-Systeme basiert auf voneinander unabhängigen Einzelarbeitsplätzen, realisiert durch einfache Mikrorechner. Im letzten Kapitel wurde angeführt, daß zum einen diese Architektur das Gefühl der Isolation bei Lernenden fördert und daß zum anderen Lernformen, wie z.B. Gruppenarbeit hierdurch unzureichend unterstützt werden. Neuere Formen des CUU, die den Zugriff auf regionale oder internationale Informationssysteme erfordern, sind durch diese Architektur nicht möglich. Um solche weitergehenden Anwendungen innerhalb von CUU-Systemen realisieren zu können, ist vom bisherigen Einzelarbeitsplatz-Ansatz abzugehen und eine erweiterte Architektur vorzusehen. Die dazu nötigen technischen Voraussetzungen liegen durch Rechnernetze und -hierarchien vor, entsprechende Vorschläge wurden für den CUU-Bereich gemacht und in Einzelfällen auch realisiert.

So schlägt Birster /Birster84/ eine Rechner-Konfiguration für einen computerunterstützten Unterricht vor, bei dem jedem Schüler ein Rechner mit externem Speicher, Bildschirm und Tastatur und dem Lehrer ein weiterer Rechner mit zusätzlichen Diskettenlaufwerken und einem Matrixdrucker zur Verfügung steht. Sämtliche Rechner werden über einen Netzwerk-Controller miteinander verbunden, wobei der Lehrerrechner als Zentrale fungiert und es ihm ermöglicht, *"sich in das Schülersystem einzuschalten, denn nur ein Lehrer kann helfend eingreifen, wenn es darum geht, den Unterrichtsstoff zu verstehen"* (Birster84/, S. 6).

Horlacher /Horlacher82/ sieht ein sogenanntes "Telesoftwaresystem" vor, das einen Großrechner mit beliebigem Standort über Bildschirmtext mit einem Schülercomputer (Mikrorechner) verbindet und so die Möglichkeit schafft, Informationen und zentral erarbeitete Unterrichtsprogramme vom Großrechner auf den Mikrorechner zu übertragen.

Ausgehend von den Anforderungen heute denkbarer CUU-Formen erscheint für einen allgemeinen Dialograhmen als Architektur eine Kombination dieser Ansätze sinnvoll und auch unter Kostengesichtspunkten mittelfristig realisierbar. Basis der hier vorgeschlagenen Architektur für einen allgemeinen Dialograhmen ist ein **lokales Mikrorechnernetz**. Ein solches Netz ist aufzubauen aus einem Klassensatz von Mikrorechnern, jeweils mit diversen Eingabegeräten, einem hochauflösenden Bildschirm und eigenem Diskettenspeicher, einem gemeinsamen Massenspeicher, sowie einem oder mehreren grafikfähigen Druckern. Dieses Netz, in das auch der gleichberechtigte Lehrerrechner integriert ist, bietet die Möglichkeit zu Einzelarbeitsplätzen, zu Kommunikation zwischen Rechnern und auch zu Zugriff auf gemeinsame Massenspeicher und unterstützt damit Lernformen wie individuelles Lernen, Gruppenarbeit, Aufbau projektbezogener Datenbestände und auch das Einschalten des Lehrers in einzelne Arbeitsgänge. Allerdings wollen wir hier nicht dem Lehrer - wie Birster (s.o.) - eine Sonderrolle einräumen. "Helfende Eingriffe" sind im persönlichen Gespräch leichter und besser vorzunehmen. Das Gefühl bei Schülern, in ihrem Mensch-Rechner-Dialog unbemerkt "beobachtet" werden zu können, dient nicht einem angstfreien Lernen.

Die in einer Schule installierten "Klassen-Netze" sind zu verbinden mit dem **übergeordneten Schulrechner**. Dieser Rechner - z.B. auch ein Mikrorechner - kann als Zentrale für die einzelnen Netze fungieren. Innerhalb der **Hierarchie Schülerrechner - Schulrechner** kommen dem Schulrechner vor allem zwei Funktionen zu. Erstens dient er dem Schüler als schulinternes Informationssystem, auf das bei Bedarf zurückgegriffen werden kann oder das von sich aus aktuelle Informationen an die Schüler weitergibt; zweitens bildet er eine Zwischenstation, die die lokalen Netze mit überregionalen Rechnern verbindet.

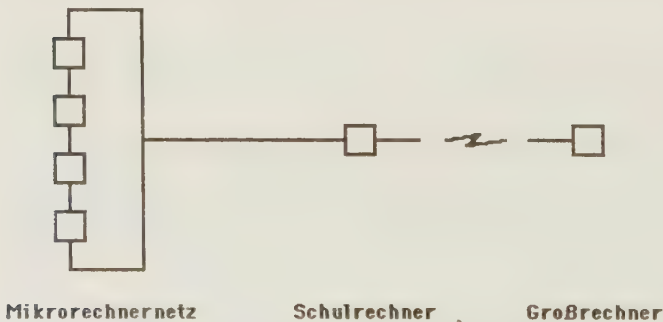


Abb. 6.2.: Architektur eines Allgemeinen Dialograhmens

Dafür ist der Schulrechner (z.B. über Bildschirmtext) mit anderen Rechnern oder Rechnernetzen zu koppeln und somit eine **dritte Ebene innerhalb der Architektur** eines allgemeinen Dialograhmens einzurichten. Diese ermöglicht jedem einzelnen Schüler den Zugriff auf beliebige Datenbanken und Informationssysteme und eröffnet weitreichende Möglichkeiten zur Computerunterstützung im Unterricht. Es sei aber angemerkt, daß die finanziellen und organisatorischen Bedingungen für die Realisierung solcher Konzepte im öffentlichen Schulsystem derzeit nicht gegeben sind.

Mit dieser Hierarchie von Rechnern gehen wir wesentlich über die Forderungen und Ansätze anderer Arbeiten hinaus (/Stork85/, /Coutaz85/, /Bullinger83/, /Balzert86/). Balzert vergleicht einige Architekturen für Systeme, die zwischen Benutzerschnittstelle und Anwendungssystem trennen und kommt mit seiner Architektur zu ähnlichen Ergebnissen, die wir - bezogen auf die Rechnerhierarchie - in den Kapiteln 12 - 14 herleiten.

7. Zur abstrakten Dialogstruktur von CUU-Systemen

Wir wollen auf dieser Betrachtungsebene zunächst einige Kritikpunkte zum Dialogverhalten von CUU-Systemen vorbringen und dann die im dritten und vierten Kapitel vorgenommenen Klassifikationen unter dem Aspekt der Benutzerfreundlichkeit für diese Ebene auswerten. Dazu stellen wir allgemein die definierten Dialogformen und -typen den Forderungen zur Benutzerfreundlichkeit (vgl. Kapitel 2) gegenüber und stellen eine "Eignungstabelle" für Dialogformen, Dialogtypen und Benutzertypen auf. Diese allgemeinen Erkenntnisse verarbeiten wir dann im dritten Abschnitt, indem wir benutzerfreundliche abstrakte Dialogstrukturen für CUU-Systeme entwickeln.

7.1. Zum Dialogverhalten von CUU-Systemen

Legt man den Aspekt der Initiative zur Charakterisierung von Dialogverhalten zugrunde, so unterscheidet man allgemein den **benutzer-geführten** und den **rechner-geführten Dialog**. CUU-Systeme sind rechner-geführt. Das Programm präsentiert dem Benutzer den Stoff, stellt Fragen, bestimmt die Reihenfolge: der Benutzer reagiert nur, er wird gesteuert. *"Zwar schätzen es die Menschen, hin und wieder angetrieben zu werden, vor allem wenn die Tätigkeit Freude bereitet (man beobachte nur einen Menschen am Flipperkasten). Aber auch dabei behält sich der Mensch gerne das Recht vor, zu beliebigem Zeitpunkt aufzuhören oder die Steuerung zu übernehmen. Und wenn die Tätigkeit mühselig ist, oder gar abgelehnt wird, dann kann die Kontrolle von außen heftige Abwehrgefühle hervorrufen."* (/Nievergelt82/, S. 10)

Die Möglichkeiten, in CUU-Systemen die Steuerung zu übernehmen, sind für den Benutzer zur Zeit nahezu nur dadurch gegeben, daß er jederzeit die Maschine abschalten kann. Die Unterbrechungsfunktion ist nur selten realisiert. Das Gefühl *"jederzeit das Steuer in der Hand zu haben"* (/Nievergelt82/), der "Herr über die Maschine zu sein", kann bei solchen geringen Funktionen beim Benutzer nicht aufkommen. Weiter ist anzumerken, daß dem Benutzer von CUU-Systemen neben der Steuerung auch weitgehend die Verantwortung für "richtiges" Handeln aus der Hand genommen wird. Die abstrakte Dialogform "Kommando" findet man nur selten, stattdessen wird fast ausschließlich das "Auswahlangebot" verwendet. Kann man durch Konkretisierungs- und Realisierungsmethoden das Menü besonders gegenüber dem gelegentlichen Benutzer als benutzerfreundlich einstufen, so ist die Ja-Nein-Frage wegen ihrer erheblichen Umständlichkeit und den mit ihr für den Benutzer verbundenen Einschränkungen (bis auf Sonderfälle) abzulehnen (vgl. den nächsten Abschnitt). Die Rechnerführung ist hier dem Benutzer unangenehm, Eigenverantwortlichkeit auf ein Minimum reduziert.

Dies gilt genauso für die syntaktischen Einschränkungen bei Datenabfragen. Fragen wie "Bitte Datum eingeben (TT,MM,JJJJ)" oder "Bitte Uhrzeit angeben (HH,MM,SS)" werden vom Benutzer genauso einschränkend empfunden, wie sie sind. Aus programmtechnischen Gründen sind sie darüberhinaus überflüssig, weil entsprechende Auswertungsprogramme nahezu jede Art der Datums- und Zeiteingabe interpretieren können (vgl auch das Beispiel in Abschn. 3.3). Fragen dieser Art lassen bestenfalls Rückschlüsse auf die Bequemlichkeit des Programmautors zu. Zu fordern ist hier, daß die Kriterien Initiative und Eigenverantwortlichkeit bei der Konzeption einer abstrakten Dialogstruktur angemessen berücksichtigt werden.

CUU-Systeme verwenden als abstrakte Dialogstruktur durchweg die Kombination unterschiedlicher Dialogformen. Der Wechsel von Dialogform zu Dialogform ist hier durch die Art der Systemausgabe erkenntlich; mißverständliches Systemverhalten in dieser Hinsicht ist selten. Nicht ganz so gut steht es mit dem Anzeigen von Möglichkeiten zum Initiatorwechsel. Gibt es innerhalb eines Systems Funktionen wie "Unterbrechen" oder "Halt", so reicht es nicht aus, einmal an zentraler Stelle darauf hinzuweisen, sondern der dem Benutzer zur Verfügung stehende Vorrat sollte permanent oder zumindest immer dann, wenn Eingaben von ihm erwartet werden, angezeigt werden.

7.2. Abstrakte Dialogstruktur, Dialogtypen und Benutzertypen

Die Kritik am Dialogverhalten von CUU-Systemen zu diesem Aspekt richtete sich in erster Linie gegen die ausschließliche Verwendung von Dialogformen mit Systeminitiative und geringer Verantwortung für den Benutzer sowie die unzureichenden Möglichkeiten zur Kombination von Dialogtypen zu einer Dialogstruktur. Um in dieser Hinsicht Verbesserungen - und damit mehr Benutzerfreundlichkeit von CUU-Systemen - zu erreichen, müssen Forderungen und Vorschläge zur abstrakten Dialogstruktur ausgehen von den verschiedenen Benutzertypen und auch den unterschiedlichen Dialogtypen.

Generell ist zu sagen, daß innerhalb von Mensch-Rechner-Dialogen mit wachsender Erfahrung der Benutzer die Initiative und Verantwortung des Systems zurückgehen muß. Befindet sich ein gelegentlicher Benutzer im Dialog mit dem Rechner, so sollte er maßvoll - z. B. durch Auswahlangebote - durch das System geführt werden, um die Möglichkeiten und die Arbeitsweise des Rechners kennen und benutzen zu lernen. Ein erfahrener Benutzer will selbst initiativ werden und kann Verantwortung übernehmen, um mit geringstem Aufwand eine Aufgabe zu erledigen. In Anlehnung an /Cakir83/ und /Dehning78/ wollen wir die Beziehungen zwischen den im dritten Kapitel eingeführten abstrakten Dialogformen, den Benutzertypen (vgl. Kapitel 2) und den Dialogtypen (vgl. Kapitel 4) folgendermaßen charakterisieren:

	Datenabfrage	Auswahlangebot	Kommando	Unterbrechung
Initiative Eigenverantwortlichkeit	System mittel	System gering	System hoch	Benutzer hoch
Anwendungs- dialog				
geeignet		gelegentlicher Benutzer	geübter Benutzer	alle Benutzer
begrenzt geeignet	alle Benutzer	geübter Benutzer		
ungeeignet			gelegentlicher Benutzer	
Kontroll- dialog				
geeignet		gelegentlicher Benutzer	geübter Benutzer	alle Benutzer
begrenzt geeignet	alle Benutzer	geübter Benutzer		
ungeeignet			gelegentlicher Benutzer	
Gestaltungs- dialog				
geeignet		gelegentlicher Benutzer	geübter Benutzer	alle Benutzer
begrenzt geeignet	alle Benutzer	geübter Benutzer		
ungeeignet			gelegentlicher Benutzer	
Meta- dialog				
geeignet		gelegentlicher Benutzer	geübter Benutzer	alle Benutzer
begrenzt geeignet	alle Benutzer	geübter Benutzer	gelegentlicher Benutzer	
ungeeignet				

Tab. 7.1.: Eignungstabelle: Benutzertypen, Dialogtypen, Dialogformen

Systeminitiierte **Datenabfragen** sind zwar unflexibel und verlangsamen einen Dialog, lassen sich aber nicht vollständig vermeiden. Nötig sind sie immer dann, wenn für eine Eingabe ein theoretisch unbegrenzter Wertevorrat in Frage kommt. Im Kontrolldialog ist dies der Fall, wenn zum Abspeichern von Daten ein Dateiname eingegeben werden muß, im Gestaltungsdialog u.a. beim Setzen von Seitenzahlen für einen Text usw. Wegen dieser Eigenschaft ist die Datenabfrage als begrenzt geeignet für alle Benutzertypen einzustufen (vgl. Tab.7.1.).

Eignungsunterschiede gibt es dabei hinsichtlich verschiedener Konkretisierungen dieser abstrakten Dialogform. So ist die Vorgabe einer Antwortsyntax grundsätzlich abzulehnen (vgl. letzten Abschnitt). Weiter wird der geübte Benutzer kurze Angaben - z.B. Zahlen oder kurze Zeichenketten - langen vorziehen und somit lieber Stichworte als quasi-natürlichsprachliche Äußerungen verwenden. Beim gelegentlichen Benutzer ist dies umgekehrt: Der Gebrauch einer ihm vertrauten Sprache ist ihm im Umgang mit dem ungewohnten Gerät angenehm und gibt ihm ein Gefühl von Sicherheit.

Eignungsunterschiede bezüglich der Dialogtypen sind kaum zu erkennen. Lediglich die Häufigkeit zur **Datenabfrage** wird im Metadialog gegenüber den anderen beiden Klassen gering sein. Hier spielt die Dateneingabe eine sehr untergeordnete Rolle. Ersatzformen für Datenabfragen können Auswahlangebot - wenn der Wertevorrat verhältnismäßig klein ist (5 - 10 Möglichkeiten) - und Kommando sein, bei dem Daten als Parameter direkt angegeben werden können. Dies betrifft vor allem den Anwendungsdialog, bei dem der Programmator vor dem Einsatz der Datenabfrage alternative Dialogformen auf ihre Eignung überprüfen sollte.

Prinzipiell eignet sich das **Auswahlangebot** durch seine breite Anwendbarkeit und die heutigen Konkretisierungs- und Realisierungsmöglichkeiten für den Mensch-Rechner-Dialog gegenüber allen Benutzertypen: *"An alternative technique, menus and selection by pointing, has gained acceptance among users of interactive systems."* (/Stelovsky85/, S. 109) (Vgl. z.B. die empirischen Untersuchungen in /Whiteside85/ und /Fährich85/ oder auch die Ausführungen in /Cakir83/ und /Foley84/.) Bei Übernahme der Initiative durch das System und geringer Eigenverantwortung seitens des Benutzers wird der Benutzer durch einen leichten, verständlichen Dialog sicher zu seinem Ziel geführt. Inwieweit die Systemführung und geringe Eigenverantwortlichkeit als aufdringlich bzw. einschränkend empfunden wird, hängt einerseits von verschiedenen Konkretisierungen und andererseits von den Erfahrungen und Vorkenntnissen der Benutzer ab.

Die **Ja-Nein-Frage** als Auswahl aus zwei Alternativen ist hinsichtlich der Benutzerfreundlichkeit sehr viel schlechter einzuschätzen als das Menü mit der Auswahl aus mehreren Möglichkeiten. Sie schränkt die Dispositionsspielräume des Benutzers stark ein, ist unflexibel und sollte deshalb nur begrenzt gegenüber

gelegentlichen Benutzern eingesetzt werden. Gegenüber geübten Benutzern - so unsere These - ist diese Form grundsätzlich abzulehnen.

Das **Menü** ist sehr viel flexibler und wird vom gelegentlichen Benutzer als angenehm empfunden (/Whiteside85/, /Fähnrich85/). Auch für geübte Benutzer ist diese Form des Auswahlangebots sinnvoll, sollte allerdings nicht ausschließlich oder überwiegend für den Dialog verwendet werden. Häufige Verwendung wird wegen der geringen Eigenverantwortlichkeit einerseits und der gegenüber dem Kommando "umständlichen Arbeitsweise" (vgl. /Shackel85/) andererseits auf geübte Benutzer in dem Maße frustrierend wirken, wie sie dem gelegentlichen Benutzer eine Hilfe ist. Seine Angst vor möglicherweise weitreichenden Eingabefehlern wird gemildert (vgl. /Kühn80/). Durch entsprechende Maßnahmen auf der Konkretisierungs- und Realisierungsebene kann mittels Menükonversation (und der Unterbrechung) für den Benutzer ein benutzergeführter Mensch-Rechner-Dialog geschaffen werden (vgl. die nächsten Kapitel). Aus diesen Gründen ist das Auswahlangebot für den gelegentlichen Benutzer als geeignet und für den geübten als begrenzt geeignet einzustufen, wobei Unterschiede hinsichtlich des Dialogtyps wieder nicht zu erkennen sind.

Hinsichtlich der abstrakten Dialogform **Kommando** ist die Einschätzung zur Eignung gegenüber den unterschiedlichen Benutzertypen gerade entgegengesetzt. Das Kommando ist grundsätzlich geeignet für den Dialog eines geübten und nicht bzw. begrenzt geeignet für einen gelegentlichen Benutzer. Dieses sehr flexible Mittel zur Durchführung von Dialogen bietet in seiner Form mit festgelegten Syntaxregeln dem geübten Benutzer die Möglichkeit, eigenverantwortlich und mit wenig Aufwand gewünschte Funktionen auszulösen. Durch zusätzliche, die Funktionen näher spezifizierende Parameter kann er - u.U. mit kurzen Dialogen - sein Ziel erreichen. Dabei erscheint ihm die beim System liegende Initiative kaum einschränkend. In Verbindung mit der Unterbrechung und mit den Konkretisierungs- und Realisierungsmöglichkeiten kann darüberhinaus die Bedeutung der System-Initiative weiter zurückgedrängt werden, sodaß auch hier ein benutzergeführter Dialog entsteht. Eine Konkretisierung des Kommandos "ohne" einschränkende Syntaxregeln als quasi-natürlichsprachliche Äußerung erscheint für den geübten Benutzer wegen des damit verbundenen höheren Eingabeaufwandes weniger geeignet. Allerdings lassen sich z.B. durch akustische Spracheingabe und Eingabeinterpretationen diese Nachteile weitgehend beseitigen.

Wegen der hohen Eigenverantwortlichkeit reagiert der gelegentliche Benutzer dem Kommando gegenüber unsicher: Er kennt nicht die Syntaxregeln, weiß über die Wirkung von Funktionen zu wenig und ist auch nicht über die Funktionenmengen ausreichend informiert. Insbesondere bei Kontrolldialogen können durch falsche Eingaben weitreichende Fehler entstehen, sodaß hier das Kommando - sowohl syntaktisch eingeschränkt wie nicht eingeschränkt - als

nicht geeignet zu betrachten ist. Entsprechendes gilt auch bei Gestaltungsdialogen, wenn auch die Folgen fehlerhafter Eingaben meist weniger stark und leichter zu reparieren sind. Im Falle von Metadiialogen sind solche weitreichenden Folgen nicht zu befürchten. Weiter ist hier der Funktionsumfang wesentlich geringer als in den vorgenannten Typen, sodaß der gelegentliche Benutzer auch durch Kommandos Metafunktionen "angstfrei" aktivieren kann. Bezüglich dieser Klasse ist das Kommando somit als begrenzt geeignet einzustufen.

Die Dialogform **Unterbrechen** ist Voraussetzung für die Schaffung benutzergeführter Mensch-Rechner-Dialoge. Sie hat nur eine Funktion, beinhaltet - durch die Möglichkeit zum Widerrufen - keine negativen Folgen und ist somit ohne Einschränkung geeignet für alle Benutzertypen.

Grundsätzlich sind die obigen Ausführungen auf Anwendungsdialoge übertragbar. Allerdings hat hier die Systeminitiative eine größere Bedeutung. Die Vorgehensweise zur Lösung eines Problems ist durch das System vorgegeben, allenfalls können unterschiedliche Verfahren und Detaillierungen angegeben werden. Doch auch hier gehen Programmverzweigungen auf Initiative der Anwendungssysteme zurück. Die Initiative und Eigenverantwortlichkeit des Benutzers - und damit die Dialogformen Kommando und Unterbrechen - treten gegenüber der Datenabfrage und dem Auswahlangebot in den Hintergrund.

Man kann hinsichtlich der Gestaltung von Mensch-Rechner-Dialogen die vier genannten Dialogtypen in zwei Gruppen einteilen und dabei die Verfügbarkeit - und damit auch die Bedeutung - einzelner Funktionen (innerhalb eines Dialog-Kontextes) zugrundelegen. Wir wollen unterscheiden, **primäre** und **sekundäre Funktionen** des Anwendungs-, Kontroll-, Gestaltungs- und Metadiialogs. Unter primären Funktionen wollen wir diejenigen verstehen, die innerhalb eines Anwendungszusammenhanges gültig, sinnvoll und naheliegend sind. Sekundäre Funktionen sind in einem Anwendungszusammenhang zwar ebenfalls gültig, ihr Gebrauch aber wenig wahrscheinlich, d.h. eher die Ausnahme.

Dieser Unterteilung liegt die Idee zugrunde, einem gelegentlichen Benutzer das Zurechtfinden innerhalb des Systems dadurch zu erleichtern, daß ihm nicht alle möglichen, sondern nur ein Teil der Funktionen präsentiert werden. Anstatt dem Benutzer eine Navigation durch Funktionsgruppen zuzumuten - als Beispiel sei hier der "Menübaum" genannt - oder zeilenorientiert Funktionsnamen auszugeben, von denen der Benutzer eine gewünschte auswählt, nimmt das System - abhängig vom Arbeitszusammenhang - eine Vorauswahl vor und stellt zur Auswahl nur die primären Funktionen dar. Sekundäre Funktionen erreicht der Benutzer dann z.B. durch den Menübaum. Notwendig für diese Vorgehensweise ist ein Modell vom jeweiligen Stand des Benutzerwissens, sodaß eine solche Vorauswahl ermöglicht wird und die jeweiligen primären Funktionen ermittelt werden können. Beispiele generell primärer Funktionen sind

- die **Kontrollfunktionen**: Eröffnen, Beenden und Unterbrechen von Anwendungen, Löschen, Speichern, Drucken von Daten, Bewegen eines Fensters über der virtuellen Ausgabefläche, anwendungsabhängige Kontrollfunktionen;
- die **Gestaltungsfunktionen**: Bewegen eines Darstellungsfeldes über den Bildschirm, Festlegungen zur Ausführlichkeit von Ausgaben und Verhaltensweisen bei Eingaben ("Dialogstufen", vgl. Kapitel 8), Regeln für Benutzereingaben (Definition von Abkürzungen, Zusammenfassen von Funktionen), anwendungsabhängige Gestaltungsfunktionen;
- und die **Metafunktionen**: Informationen zum aktuellen Systemzustand und zu den möglichen Funktionen, Hilfestellungen, um vom augenblicklichen in einen gewünschten Systemzustand zu gelangen, allgemeine Hilfestellung, Rücknahme des letzten Dialogschritts.

In ihrer Bedeutung nachgeordnet, teilweise anwendungsabhängig und möglicherweise erst durch Zwischenschritte aktivierbar sind alle übrigen Kontroll-, Gestaltungs- und Metafunktionen, die deswegen hier unter "sekundäre Funktionen" zusammengefaßt werden.

7.3. Eine abstrakte Dialogstruktur für den allgemeinen Dialograhmen

Kommen wir zum Schluß dieses Kapitels zur abstrakten Dialogstruktur von CUU-Systemen. Pauschal läßt sich dazu ganz einfach fordern, daß eine Kombination der oben diskutierten abstrakten Dialogformen in Abhängigkeit von den verschiedenen Benutzertypen in der Weise zu erfolgen hat, daß den zu Beginn dieses Kapitels zur Benutzerfreundlichkeit genannten Kriterien entsprochen wird. Zunächst läßt sich aus dieser Forderung folgern, daß es eine "beste" Dialogstruktur nicht geben kann, sondern Beziehungen zu Benutzertypen hergestellt werden müssen. Die Zuordnung eines Benutzers zu einem Benutzertyp wird dabei in der Hand des jeweiligen Benutzers liegen: das System - hier der allgemeine Dialograhmen - unterbreitet Vorschläge in Form von Grundversionen für Dialogstrukturen.

Wie müssen oder sollten solche Grundversionen aussehen? Wegen der obigen Ergebnisse zu den verschiedenen abstrakten Dialogtypen und den grundlegenden Forderungen nach angemessener Eigenverantwortlichkeit und Steuerbarkeit des Dialogs durch den Benutzer muß für einen allgemeinen Dialograhmen das Unterbrechen eine zentrale Stellung einnehmen. Nur hierdurch kann beim heutigen Stand der Technik ein benutzergeführter Dialog konzipiert und dem Benutzer das Gefühl vermittelt werden, "jederzeit das Steuer in der Hand zu haben".

Dazu ist das Unterbrechen innerhalb des allgemeinen Dialograhmens für jeden Benutzer stets als Funktion verfügbar; es wird verwendet, um einen Wechsel des Dialogtyps oder einer Funktion innerhalb eines Typs zu initiieren. Für die weiteren Dialogschritte sind unterschiedliche Vorgehensweisen entsprechend den Fähigkeiten der Benutzern nötig. Wir wollen zwei Extreme, die "Dialoggrundversionen" für den gelegentlichen und für den geübten Benutzer, näher erläutern. Beliebige Zwischenformen wären denkbar, werden für den Dialograhmen auf dieser Betrachtungsebene aber nicht für notwendig gehalten. Anpassungen des Dialogverhaltens an "geübtere gelegentliche Benutzer" und an "weniger erfahrene geübte Benutzer" können durch Maßnahmen auf den Betrachtungsebenen "Konkrete Dialogstruktur" und "Dialogrealisierung" erreicht werden. Es ist hier wieder auf den "dynamischen Aspekt" des Mensch-Rechner-Dialogs hinzuweisen (vgl. Kapitel 2), der es erforderlich macht, daß das Dialogverhalten des Systems sich dem lernenden Verhalten des Benutzers - der dadurch vom gelegentlichen zum geübten Benutzer wird - anpaßt.

Die abstrakte Dialogstruktur für gelegentliche Benutzer

Basis dieser Methode ist ein Wechsel zwischen den abstrakten Dialogformen "Unterbrechung" und "Auswahlangebot", also ein Initiatorwechsel nach dem Anstoßen eines Dialogs durch den Benutzer und eine Übernahme der Dialogführung durch das System. Abhängig von einzelnen Funktionen werden außer dem Auswahlangebot auch Datenabfragen zur Klärung von Funktionsparametern in die Dialogmethode aufgenommen. Vorbild für dieses Konzept ist das Dialogverhalten von Systemen wie Xerox-Star, Apple Lisa und Apple Macintosh, das heute allgemein als richtungsweisend hinsichtlich Benutzerfreundlichkeit für gelegentliche Benutzer gilt.

Im einzelnen ist die Grundidee für diese Struktur durch folgende Dialogschritte zu kennzeichnen:

- Durch die Unterbrechungsfunktion "Verlasse aktuelle Arbeitsumgebung" (in dem System Apple Macintosh realisiert durch Auslösen und Festhalten der (einen) Maus-Funktionstaste, bei XEROX-Star durch Auslösen einer der drei Maus-Funktionstasten) wechselt der Benutzer vom Dialog eines Ausgangstyps in den Dialog eines Zieltyps. Ausgangs- und Zieltyp können

dabei identisch sein, d.h. der Benutzer unterbricht die Bearbeitung einer Funktion, um eine neue zu initiieren.

- Das System reagiert auf die Aktion des Benutzers, indem es die Initiative übernimmt und dem Benutzer ein Menü mit den im Zieltyp im augenblicklichen Systemzustand möglichen Funktionen präsentiert und ihn auffordert, eine Funktion auszuwählen.
- Der Benutzer reagiert durch Auswahl einer Funktion (in dem System Apple Macintosh realisiert durch Beenden des Auslösens der Maus-Funktionstaste). Es beginnt ein rechnergeführter Dialog mit den Formen Auswahlangebot und Datenabfrage. Dieser Dialog ist abhängig von der ausgewählten Funktion und kann durch die obige Funktion wiederum unterbrochen werden (in den o.g. genannten Systemen nur ansatzweise verwirklicht).
- Nach Beendigung des Dialogs zu der ausgewählten Funktion schaltet das System in den Dialog des Ausgangstyps zurück, setzt also dort die Arbeit fort, wo sie zuvor unterbrochen wurde.

Dieser Wechsel in der Dialoginitiative gewährleistet zum einen die Steuerbarkeit des Dialogs durch den Benutzer, da dieser jederzeit eine Dialogumgebung zugunsten einer anderen verlassen kann. Zum anderen gibt er ihm durch die Menütechnik Gelegenheit, sich in dem System zurechtzufinden, sich maßvoll führen zu lassen.

Die abstrakte Dialogstruktur für geübte Benutzer

Geübten Benutzern bietet der allgemeine Dialograhmen einen Wechsel zwischen der Unterbrechung und dem Kommando für die Typen Anwendungs-, Kontroll-, Gestaltungs- und Metadialog an. Diese Vorgehensweise orientiert sich an den Kommandosprachen herkömmlicher Betriebssysteme und hat sich als grundsätzlich geeignet und wegen der obigen Bemerkungen auch als benutzerfreundlich für geübte Benutzer erwiesen. Das Apple-Macintosh-System bietet diese Struktur ansatzweise als Alternative zur Menü-Methode für geübte Benutzer an.

Die einzelnen Dialogschritte dieser Struktur stehen in folgender Beziehung zueinander:

- Durch die Unterbrechungsfunktion "Verlasse aktuelle Arbeitsumgebung" wechselt der Benutzer vom Dialog eines Ausgangstyps in den Dialog eines Zieltyps.
- Das System reagiert auf die Aktion des Benutzers durch die Aufforderung, ein Kommando einzugeben.
- Nach der Benutzereingabe führt das System die Funktion durch und ergreift gegebenenfalls die Initiative zu einem rechnergeführten Dialog, um die zur Durchführung notwendigen Daten zu erhalten. Während dieser Phase kann der Benutzer durch eine Unterbrechung jederzeit wieder die Initiative übernehmen.

- Nach Beendigung des Dialogs zu der angegebenen Funktion schaltet das System in den Dialog des Ausgangstyps zurück.

Zusammen mit Dialoggestaltungsfunktionen zum Zusammenfassen von Kommandos, Setzen und Abfragen von "Kommando-Bedingungen" resultiert aus dieser Struktur größtmögliche Freiheit und Eigenverantwortlichkeit im Gebrauch des allgemeinen Dialograhmens.

8. Zur konkreten Dialogstruktur von CUU-Systemen

Die abstrakten Dialogformen Datenabfrage, Auswahlangebot, Kommando und Unterbrechung werden konkret durch Festlegungen hinsichtlich des zu verwendenden Vokabulars, des inneren und äußeren Formats bei Ein- und Ausgaben, durch Vorgaben bezüglich formaler Redundanz und semantischer Eigenschaften von Eingaben, sowie durch Regeln für die Interpretation und inhaltliche Ausdrucksform von Ausgaben. Da es für diese Festlegungen, Vorgaben und Regeln einen theoretisch unbegrenzten Wertevorrat gibt, lassen sich für interaktive Systeme beliebig viele konkrete Strukturen bilden.

In welcher Form sich auf dieser Betrachtungsebene das Programm darstellt, hing lange Zeit wegen fehlender Untersuchungen und Erkenntnisse nur von der Vorgehensweise des Programmators ab: er legte "nach seinem Gefühl" die konkrete Dialogstruktur fest. Die Folge davon war ein unsystematisches und uneinheitliches Bild interaktiver Systeme. Fehlende Systematik und Uneinheitlichkeit sind nun nicht schon für sich allein abzulehnen - erstens können durch persönliches Engagement durchaus benutzerfreundliche Programme konstruiert werden und zweitens kann Uneinheitlichkeit auch benutzerfreundlich sein, indem es z.B. den Benutzer herausfordert, neugierig macht oder anspornt. Wenn allerdings beim Benutzer durch widersprüchliche Vorgaben innerhalb eines Systems bzw. zwischen verschiedenen Systemen Irritationen und Frustrationen erzeugt werden, so ist diese Vorgehensweise abzulehnen.

Die meisten CUU-Programme sind Beispiele für diesen zweiten Fall; die wesentlichen Kritikpunkte werden im ersten Abschnitt dieses Kapitels diskutiert. In den darauffolgenden Abschnitten wollen wir dann die heutigen Erkenntnisse zur Vorgehensweise für diese Betrachtungsebene zusammenfassen und in Form von Forderungen für CUU-Systeme detaillieren. Dadurch stellen wir den vorgeannten abstrakten Dialogstrukturen benutzerfreundliche konkrete Strukturen gegenüber.

8.1 Zum Dialogverhalten von CUU-Systemen

Für das **Vokabular von CUU-Systemen** werden ausschließlich natürlich-sprachliche Worte und Codes aus alphanumerischen Zeichen benutzt, ohne daß Piktogramme hier Verwendung finden. Diese Vorgehensweise entspricht nicht mehr dem heutigen Stand von Wissenschaft und Technik. Forschungsergebnisse aus Bereichen wie "Kognitive Psychologie", "Kognitive Ergonomie" oder auch "Software Ergonomie" (vgl. z.B. /CHI83/, /Stauffer85/ oder /van der Veer84/), sowie daraus resultierende Systementwicklungen (Xerox-Star, Apple-Lisa, Apple-Macintosh) zeigen, daß eine Integration von Codes aus

Symbolen oder grafischen Zeichen in das Vokabular eines Dialogsystems in Verbindung mit neuen Eingabetechniken ("Maus", siehe nächstes Kapitel) zu erheblich verbessertem Dialogverhalten solcher Systeme führt.

Das **innere Format der Eingabe bei CUU-Systemen** zeichnet sich durch eine erhebliche Uneinheitlichkeit aus. Die Uneinheitlichkeit besteht hier in zweierlei Hinsicht: zum einen innerhalb eines CUU-Programms, zum zweiten zwischen verschiedenen CUU-Programmen.

Innerhalb eines Programms werden in der Regel unterschiedliche abstrakte Dialogformen verwendet. Diese besitzen typische Konkretisierungsformen, z.B. Festlegung in der Reihenfolge bei einer Datenabfrage, konkretisiert als "Aufforderung mit Antwortsyntax", oder Vorschriften in der Verwendung bestimmter Zeichen bei einem "Menü" oder einer "Ja-Nein-Antwort". Insofern ist Uneinheitlichkeit bei dem inneren Format von Eingaben unumgänglich und nicht negativ zu werten. Kritik an der Uneinheitlichkeit ist aber dann z.B. berechtigt, wenn innerhalb eines Programms eine "Ja-Nein-Antwort" einmal durch "Ja" oder "Nein", einmal durch "J" oder "N", dann durch "j" und "n" und viertens durch "Y" oder "N" einzugeben ist oder wenn auf eine entsprechende Frage einmal "Ja", "Nein" und "J", "N" geantwortet werden kann und an anderer Stelle nur eine Form akzeptiert wird. Entsprechende unterschiedliche Konkretisierungen sind auch für die übrigen Dialogformen denkbar. Zu fordern ist hier, daß jede abstrakte Dialogform bei jedem Auftreten auf dieselbe Weise zu konkretisieren ist.

Werden die meisten CUU-Programme dieser Forderung im Großen und Ganzen noch gerecht, so ist Uneinheitlichkeit im inneren Format der Eingabe zwischen verschiedenen CUU-Programmen die Regel. Betroffen hiervon ist das ganze Spektrum der abstrakten Dialogtypen. Je nach Programmautor werden unterschiedliche Konkretisierungen vorgesehen. Die Ja/Nein-Alternative, ein Datum oder eine Uhrzeit sind einmal in englischer, einmal in deutscher Form einzugeben, Trennsymbol zwischen Daten ist in einem Programm das Leerzeichen, im zweiten das Komma und im dritten der Schrägstrich, usw. *"Der Befehl 'Verlasse die aktuelle Befehlsumgebung' wird immer vorhanden sein, nur heißt er einmal Q (quit) und ein anderes Mal E (end) oder X (exit). Der Benutzer wird also gezwungen, für jedes Programm die zugehörige Befehlsumgebung wenigstens teilweise auswendig zu lernen. Eine Konvention, die wenigstens die Bezeichnung allgemeiner Befehle zur Dialogsteuerung regelt, würde dem Benutzer das Zurechtfinden in interaktiven Programmen erleichtern."* (Nievergelt83/, S. 31/)

Neben der Uneinheitlichkeit im inneren Format von Eingaben sind zu diesem Kriterium weiter Fragen erwähnenswert, auf die wir jedoch nicht näher eingehen wollen: Sind einige grammatische Regeln anderen vorzuziehen? Sind Vorschriften zur Verwendung bestimmter Zeichen notwendig, oder lassen sich nicht auch durch programmtechnische Mittel größere Freiheiten für den Benutzer erreichen? Sollte nicht jeder Benutzer "sein inneres Format der Eingabe" selbst

wählen können und das Programm dann sich dem Benutzer anpassen? (Vgl. hierzu auch /Nievergelt83/, S. 31)

Zum **äußeren Format der Eingabe bei CUU-Systemen** ist zu sagen, daß meist eine bestimmte Position auf dem Bildschirm für die Eingabe vorgesehen ist (nach einem "Cursor") und daß für gewöhnlich die Eingabe durch ein Eingabe-Abschlußzeichen zu beenden ist. Hinsichtlich dieses Zeichens herrscht wieder allgemeine Uneinheitlichkeit. So findet sowohl das Zeichen (die Taste) RETURN als auch das Zeichen (die Taste) CR (Carriage Return) Verwendung - dies ist Tastatur-abhängig und betrifft somit die vierte Betrachtungsebene des Dialogs (s.u.). Bei einer Aufforderung, nur ein Zeichen einzugeben, fehlt auch noch das Return. Daneben findet man aber auch oft in CUU-Systemen die Aufforderung, mit einer beliebigen Taste den Programmablauf weiterzuführen.

Eine **formale Redundanz der Eingabe bei CUU-Programmen** in der Form, daß Füllwörter oder zusätzliche Leerzeichen vom Programm toleriert, d.h. als solche erkannt werden, ist nur in ganz geringem Umfang möglich. Nur wenn quasi-natürlichsprachliche Eingaben erwartet werden, kann der Benutzer seine Antwort formal redundant gestalten. Das Programm sucht dann in der Antwort nach bestimmten Schlüsselwörtern und ignoriert die "überflüssigen" Füllwörter. Es wäre hier zu fordern, dem Benutzer entsprechende Freiheiten auch für andere Dialogformen zur Verfügung zu stellen. Da die Suche häufig nur kontextfrei durch einfachen Stringvergleich erfolgt, gibt es viele Möglichkeiten der Fehlinterpretation (Beispiel: C: Nenne die Hauptstadt von Italien! L: Bromberg C: Richtig!).

Das **Format der Ausgabe bei CUU-Systemen** ist sehr vielfältig. Hier wird das gesamte Spektrum von vollständigen Sätzen über Stichwörter, Formeln und grafischen Darstellungen bis hin zu Tabellen eingesetzt. Kritisch anzumerken ist hier daß dieser Einsatz mehr oder weniger konzeptlos stattfindet.

Jeder Programmautor wählt die nach seinem Geschmack geeignete Ausgabeform, wobei oft die Einfachheit der Implementierung eine entscheidende Rolle spielt. Textausgaben sind leichter in einem Programm zu realisieren als Tabellen oder grafische Darstellungen. Folge davon ist, daß vollständige Sätze und Stichwörter als Ausgabeformat überwiegen und grafische Darstellungen nur dort verwendet werden, wo sie zur Beschreibung eines Sachverhalts unumgänglich sind. Formeln werden in der Regel zeilenorientiert ausgegeben; dies ist eben auch leichter zu implementieren als eine strukturierte Darstellung z.B. mit Bruchstrichen.

Der Benutzer ist bei dieser Vorgehensweise der Leidtragende. Er muß zum Teil sehr lange verbale Beschreibungen durcharbeiten (z.B. in Simulationsprogrammen), hat, wenn er endlich zur Durchführung einer Aufgabenstellung (z.B. einer Simulation) aufgefordert ist, Teile der Beschreibung wieder vergessen, macht daraufhin Eingabefehler oder reagiert verunsichert. Möglichkeiten,

vorangegangenen Text wieder auf den Bildschirm zu holen oder - wie Nievergelt fordert - Beschreibungen gleichzeitig auf Bildschirm und Papier auszugeben, sind nicht vorgesehen. Generell ist hier die Entwicklung eines allgemeinen Konzepts zu fordern, durch das wissenschaftlich begründete Festlegungen getroffen werden, welche Ausgabeformate zu welchem Zweck geeignet und zu verwenden sind.

In Bezug auf **die semantischen Eigenschaften von Eingaben in CUU-Systemen** ist generell ungenügende Flexibilität festzustellen. Plausibilitätsüberprüfungen sind selten vorhanden und wenn, dann nur in einfachster Form: So wird zwar z.B. bei der Auswahl einer Menüalternative festgestellt, ob der eingegebene Wert innerhalb des vorgesehenen Bereichs liegt oder - falls nicht - eine Wiederholung der Eingabe verlangt. Geht es aber um Eingaben für Berechnungen z.B. bei Simulationen, erfolgt meist keine Kontrolle. Dies führt dazu, daß z.B. unsinnige Simulationen berechnet werden, gespeicherte Daten fehlerhaft sind, oder daß es zu rechnerbedingten Programmabbrüchen kommt - meist mit Betriebssystem-Fehlermeldungen (in der Regel in Englisch) -, weil eingegebene Daten vom Typ oder von der Größe nicht den zulässigen entsprachen. Ein anderes Beispiel für ungenügende Systemflexibilität ist, daß unterschiedliche Eingabemöglichkeiten richtiger Antworten oft nicht berücksichtigt werden (Beispiel: C: Wieviel ergibt $2+2$? L: Vier C: Falsch (Weil "4" als "richtige" Eingabe erwartet wurde; vgl. Abschnitt 6.2); oder: C: Nenne eine alte Hafenstadt im nordöstlichen Italien. L: Venezia C: Falsch (Weil "Venedig" als "richtige" Eingabe erwartet wurde)).

Ähnlich verhält es sich mit widersprüchlichen oder überbestimmten Antworten. Diese sind vom Programm her nicht vorgesehen und werden nicht verarbeitet. Stattdessen erfolgt - soweit der "Fehler" im Programm "vorgesehen" ist (s.o.) - die Ausgabe einer Fehlermeldung mit der Aufforderung, Werte nochmals einzugeben. (Mitunter enthalten Aufforderung und Fehlermeldung auch eine Erklärung des vorangegangenen Fehlers mit einem Beispiel, wie die Eingabe richtig zu tätigen ist.)

Dieses letzte Beispiel ist dann als eine **Interpretationsvorschrift der Ausgabe innerhalb eines CUU-Programms** zu werten. Solche Vorschriften sind in CUU-Systemen allerdings nicht die Regel. Oft werden Ausgaben nur im Dialogkontext verständlich, bzw. es bedarf eines Nachschlagens in Systembeschreibungen (vgl. die Problematik der Unsicherheit bei Werteingaben für Simulationen aufgrund vergessener oder überlesener Ausgaben (s.o.)). Sicherlich kann bei Ausgaben, die eine Aufforderung zur Eingabe beinhalten, nicht in jedem Fall eine vollständige Interpretationsvorschrift geliefert werden. Hier ist ein praktikabler Kompromiß zwischen Selbsterklärung und Interpretation im Kontext - bzw. durch zusätzliche Information - notwendig, damit der Zweck von CUU-Programmen nicht verlorengeht.

Für die übrigen Ausgaben dagegen ist auf eine weitgehende Verständlichkeit auch ohne zusätzliche Information nicht zu verzichten. Wenn also Bezug genommen wird auf vorherige Ausgaben, die nicht mehr auf dem Bildschirm zu sehen sind, so müssen die notwendigen Teile noch einmal dargestellt werden, bzw. es muß technisch möglich sein, auf solche Teile erneut zuzugreifen (vgl. den nächsten Abschnitt). Weiter sind Stichwörter selbsterklärend zu wählen; Zahlenangaben müssen mit erläuternden Texten (z.B. Maßeinheiten) versehen sein.

Der letzte hier zu behandelnde Aspekt ist die **inhaltliche Ausdrucksform der Ausgabe in CUU-Programmen**. Das Verhalten von CUU-Systemen unter dieser Sichtweise ist wiederum geprägt von den Intentionen der jeweiligen Programm-Entwickler. Abhängig von deren Einstellungen, Erfahrungen und Zielen sind lange, ausführliche, kurze und knappe Ausgaben zu finden.

An welchen Stellen Meldungen ausführlich oder weniger ausführlich notwendig sind, hängt einerseits vom bearbeiteten Problem ab und ist - gerade bei Lernprogrammen - eine methodische Frage. Wir wollen hierauf nicht näher eingehen und nur darauf hinweisen, daß die für diesen Bereich vorliegenden Erkenntnisse von Programmautoren zu berücksichtigen sind (man denke z.B. an die Vorgehensweise, je nach Erfahrung des Benutzers unterschiedlich ausführliche Ein- und Ausgabemöglichkeiten vorzusehen (vgl. z.B. /Kühn80/)) und diese nicht so etwas wie den "Implementierungsaufwand" als Kriterium für die inhaltliche Ausdrucksform der Ausgabe zugrundelegen dürfen. Ferner bedarf es auch hier wieder grundlegender allgemeingültiger Konzepte, um die Programmautoren entsprechend zu unterstützen.

In diese Konzepte müssen natürlich auch psychologische Aspekte eingehen, wie durch Ausgabe-Gestaltung Motivation gefördert und Angst beseitigt werden kann. Sicherlich wenig geeignet sind in diesem Zusammenhang beleidigende und freche Fehlermeldungen: "Was soll das, kannst du nicht richtig antworten????? Also nochmal!"

8.2. Piktogramme

Zum **Vokabular von CUU-Systemen** wurde die Bedeutung von Piktogrammen in Abschnitt 3.3. bereits angedeutet. Durch die Tatsache, daß der Mensch einfache grafische Symbole leichter aufnimmt und mit einem Sachverhalt assoziiert, als dies bei Texten und Beschreibungen der Fall ist, werden Piktogramme heute überall eingesetzt: als Verkehrszeichen, in Druckwerken und auf technischen Zeichnungen, auf Straßen, Bahnhöfen, Flughäfen, in größeren Gebäuden.

Innerhalb interaktiver Computer-Systeme fanden Piktogramme zuerst in der grafischen Datenverarbeitung - etwa bei CAD-Systemen - Verwendung, z.B. zur

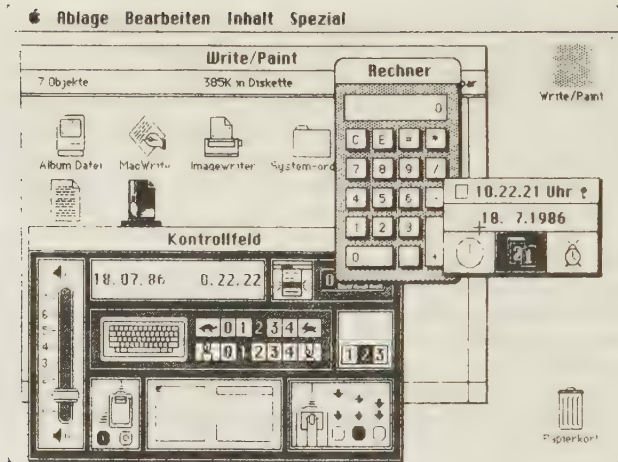


Abb. 8.2: Beispiele für Piktogramme und Fenster des Apple-Macintosh

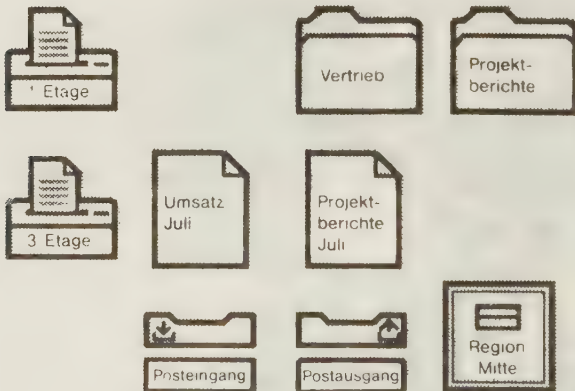


Abb. 8.3: Beispiele für Piktogramme des Xerox-Star /Rank Xerox85/

Eine Sonderrolle kommt hier der Unterbrechungsfunktion zu. Sie ist nicht unbedingt durch ein eigenes Piktogramm zu konkretisieren, sondern kann auch in andere Funktionen integriert werden. Eine Auswahl einer beliebigen Funktion.

z.B. durch Piktogramm, beinhaltet dann das Unterbrechen der Anwendung. Auf diese Weise wird der Wechsel der Initiative vom Unterbrechen zum Menü verschleiert. Der Benutzer wählt direkt die im Menü enthaltene Funktion und fühlt sich trotz System-Initiative als Dialogführer. Die sekundären Funktionen der Dialogtypen können dann jeweils zu Gruppen zusammengefaßt werden. Durch Aktivierung eines die Gruppe kennzeichnenden Piktogramms werden sie mit Hilfe von Menüs aufgerufen ("Menübaum", "Menühierarchie").

Geübten Benutzern wird diese Konkretisierung nicht gerecht. Für sie ist die Verwendung von Piktogrammen in den Hintergrund zu rücken. Stattdessen sollte wahlweise der Kommandodialog durch natürlich-sprachliche Ein- und Ausgabe konkretisiert werden. Hierfür bietet es sich an, entsprechend den Dialogtypen unterschiedliche Steuerzeichen (Sondertasten) zur Einleitung vorzusehen und den Aufruf von zur Verfügung stehenden Funktionen durch Festlegung selbsterklärender Kommandonamen zu erleichtern. Auch hier kann das Unterbrechen und die Eingabe eines Kommandos in einem Schritt erfolgen. Das Steuerzeichen gibt dabei die Funktion "Verlasse aktuelle Arbeitsumgebung" wieder und spezifiziert gleichzeitig die gewünschte Zielklasse. Der Dialog erfolgt dann benutzergesteuert. Die Unterscheidung von primären und sekundären Funktionen spielt hier nur eine untergeordnete Rolle. Da der geübte Benutzer mittels Kommando eine gewünschte Funktion auswählt, wird ihm der Funktionsvorrat nur bei Bedarf (z.B. durch Aufruf einer Metafunktion) präsentiert. Hier kann dann die Unterteilung sinnvoll sein.

8.3. Cursor und Fenster

Zum Format der Eingabe wie auch der Ausgabe wurde die vorherrschende Uneinheitlichkeit kritisiert. Dazu ist hier anzumerken, daß bereits durch das Konzept des allgemeinen Dialograhmens mit der Einbettung von Anwendungsfunktionen in die Dialogtypen weitgehend einheitliche Arbeitsweisen erreicht werden.

Wichtig sowohl zur Interpretation von Ausgaben wie auch für das äußere Format von Eingaben ist ein Systemverhalten, das dem Benutzer den jeweiligen Systemzustand und den Arbeitszusammenhang transparent macht. Wählt man je nach Dialogtyp unterschiedliche Cursor und kombiniert deren Darstellung mit der Fenstertechnik, so kann der geübte Benutzer Aus- und Eingaben einschätzen, fühlt sich der gelegentliche Benutzer sicher. Dazu ist jedem Dialogtyp ein Fenster zuzuordnen, das die zugehörigen Dialogschritte dokumentiert. Die über Tastatur getätigten Eingaben eines geübten Benutzers werden hier wiedergegeben, ebenso Systemausgaben. Die Bereitschaft, neue Kommandos zu empfangen, bzw. die Aufforderung, Eingaben zu tätigen, wird innerhalb des Fensters durch die jeweiligen Cursor angezeigt.

Dabei kann dieses Fenster dem geübten Benutzer zeilenorientiert präsentiert werden. Wichtig ist dann, daß das System tolerant bzgl. formaler Redundanz und gegenüber Syntaxfehlern ist. "J", "j", "JA", "ja", "JAWOHL" oder "jawohl" sind dann alles zulässige Antworten auf eine Ja-Nein-Frage. Zwei Wege für ein benutzerfreundliches Systemverhalten hinsichtlich der Fehlertoleranz sind möglich. Zum einen kann das System eine fehlerhafte Eingabe ablehnen und den Benutzer auffordern, den Fehler zu korrigieren. Dazu gibt ihm das System Hilfe, indem es anzeigt, wo ein Fehler ist und um welche Art Fehler es sich handelt. Es kann auch ein Vorschlag für die Korrektur gegeben werden. Eine Interpretation solcher Fehlermeldungen muß ohne Nachschlagen in Handbüchern möglich sein. Zum zweiten besteht die Möglichkeit, daß das System auf der Grundlage des Systemzustandes versucht, den Fehler selbstständig zu korrigieren. Wegen der möglichen Fehlinterpretationen durch das System sollte die korrigierte Version dem Benutzer dann zur Zustimmung präsentiert werden.

Dem gelegentlichen Benutzer gegenüber muß ein Fenster grafisch aufgearbeitet werden (Das Apple-Macintosh-System kann hier durchaus als Vorbild herangezogen werden; vgl. auch die Dialogrealisierung.) Die grafische Präsentation führt dann dazu, daß die Beantwortung einfacher Anfragen und Ja-Nein-Fragen sich von der Auswahl von Menüalternativen und dem Initiieren eines Kommandos nur geringfügig unterscheidet. Dem gelegentlichen Benutzer wird der Unterschied kaum noch bewußt; ihm gegenüber ist ein transparentes, leicht erlernbares und verständliches Systemverhalten erreicht. Die Probleme von formaler Redundanz und Interpretationsvorschriften von Ausgaben stellen sich dann in der kritisierten Form nicht mehr. Der Benutzer hat darüberhinaus die Entscheidung, wann er seine Eingaben dem System übergeben will, kann also Tippfehler oder Versehen vorher ohne Folgen und komplizierte Aktionen beseitigen.

Bei verbleibenden Situationen, in denen Syntaxfehler möglich sind, ist entsprechend fehlertolerantes Systemverhalten zu fordern. Versucht werden sollte, hier zusätzlich fehlertolerantes Verhalten gegenüber semantischen Fehlern zu erreichen. Dazu bedarf es einer System-Wissensbasis, die aufgrund vorheriger Dialogschritte Plausibilität von Eingaben überprüft und in Zweifelsfällen den Benutzer auf mögliche Inkonsistenzen anspricht. Die Angst vor folgenschweren Fehlern wird dadurch beim Benutzer weiter reduziert.

8.4. Dialogstufen

Wie bei der Beschreibung der abstrakten Dialogstruktur erwähnt, erreicht man unterschiedliche Anpassungen des Systems an unterschiedliche Fähigkeiten von Benutzern u.a. durch Maßnahmen auf der Konkretisierungsebene. Eine Methode hierzu sind verschiedene Dialogstufen. Vom Benutzer durch eine Gestaltungsfunktion eingestellt, präsentiert das System einmal umfangreiche

Beschreibungen und Fehlermeldungen, gibt detailliert Auskunft über Möglichkeiten und Erweiterungen; ein anderes Mal werden Abkürzungen und Schlüsselwörter verwendet, oder es verzichtet drittens ganz auf Beschreibungen. Solche Dialogstufen zur Ausführlichkeit von Ausgaben sind im allgemeinen Dialograhmen vorzusehen.

Bei Eingaben sollten unterschiedliche Dialogstufen fließend ineinander übergehen. D.h., dann entscheidet der Benutzer, ob er Piktogramme und grafisch aufbereitete Fenster verwendet oder ob er Kommandos mit ausführlicher oder abgekürzter Parameterbeschreibung benutzt. Jede dieser Eingabeformen sollte vom System verarbeitet werden. Solche Eigenverantwortlichkeit des Benutzers fördert die Zufriedenheit und Sicherheit und das "allgemeine Wohlbefinden" (vgl. /Kühn80/) und bietet damit ein hohes Maß an Benutzerfreundlichkeit.

Ein "lernendes Verhalten" des Systems, das in Abhängigkeit vom Benutzerverhalten selbstständig zwischen Dialogstufen umschaltet, ist ebenfalls denkbar. Die heutigen Entwicklungen zur Wissensanalyse werden in dieser Richtung weiter vorangetrieben. Nicht ganz unproblematisch in diesem Zusammenhang ist die "Beobachtung" des Benutzerverhaltens durch das System. Das Gefühl, ständig beobachtet zu werden, verunsichert u.U. den Benutzer. Aktive Hilfesysteme - wie Bauer /Bauer85/ sie fordert - dürfen nicht aufdringlich sein, sondern sollten den Benutzer nur anleiten, wenn dieser es ausdrücklich verlangt. Mit zunehmender Übung erkennt der Benutzer selbst die Umständlichkeit mancher seiner Tätigkeiten und wird von sich aus Vereinfachungsmöglichkeiten wünschen. Eine Aktivität des Systems in der Form, daß das System selbstständig nach wiederholten umständlichen Eingaben Erleichterungen vorschlägt, ist unseres Erachtens wenig benutzerfreundlich. Es fördert das Gefühl beobachtet zu werden und sorgt für Unsicherheit bei nachfolgenden Eingaben: "Soll ich nun trotz Ermahnung die mir geläufige einfache Form wählen oder soll ich - mit der Gefahr, etwas falsch zu machen - die noch nicht ganz bekannte neue Version benutzen". In diesem Sinne muß das "Sich Anpassen" des Systems schrittweise erfolgen, auf eine Weise, daß der Wechsel von Dialogstufen unmerklich vor sich geht.

9. Zur Dialogrealisierung von CUU-Systemen

Ausgangspunkt für die folgenden Betrachtungen sind die im Abschnitt 3.4. vorgestellten Begriffe und Techniken zur Realisierung von Mensch-Rechner-Dialogen. Nach einer kurzen Kritik am Dialogverhalten von CUU-Programmen werden zunächst ohne Berücksichtigung abstrakter Ein- und Ausgabefunktionen auf sie wird dann im nächsten Kapitel eingegangen - Ein- und Ausgabegeräte, Fenstertechnik und Darstellungstechniken in Beziehung gesetzt zu den oben entwickelten abstrakten und konkreten Dialogstrukturen. Dies erfolgt in Unterscheidung von gelegentlichen und geübten Benutzern. Zwischenformen bei der Realisierung beschreiben wir dann im fünften Abschnitt dieses Kapitels.

9.1. Zum Dialogverhalten von CUU-Systemen

Auch auf dieser Betrachtungsebene des Dialogs ist für CUU-Systeme kein methodisches Vorgehen erkennbar. Mit welchen Mitteln man Mitteilungen in einer Mensch-Rechner-Umgebung verwirklichen kann, bzw. sollte, ist in der Theorie zwar heute oft diskutiert, auch durch vielversprechende Ansätze gekennzeichnet (vgl. die Veröffentlichungen der Fachgruppe "Interaktive Systeme" der Gesellschaft für Informatik, Tagungen zur Software-Ergonomie und zur Kognitiven Ergonomie), doch ist diese Entwicklung bisher an CUU-Systemen nahezu spurlos vorübergegangen. Die Gründe hierfür sind vielfältig:

- Fehlende finanzielle Mittel erlauben keine Ausstattung von Schulen mit modernen Rechnern.
- Einzelengagement von Firmen, Schulen und Elternschaften führt zu einer enormen Gerätevielfalt, die einheitliche Vorgehensweisen bei der Softwareentwicklung behindert.
- Konzeptlosigkeit im Bereich CUU zwingt jeden Programmautor, der technisch und theoretisch in der Regel nicht auf dem neuesten Stand ist, "vor sich hin zu werkeln" und die seiner Meinung nach geeignete Realisierung zu wählen. Die Programmautoren sind häufig isoliert arbeitende informatische Autodidakten (wohl aber Experten in ihrem Fach) ohne Kenntnisse in Software Engineering und Software Ergonomie.

Dies führt dann dazu,

daß CUU-Programme Möglichkeiten zur Bildschirmstrukturierung (Fenster und Darstellungsfelder) nicht nutzen. Sie arbeiten zeilenorientiert mit einfachen WRITE Befehlen. Dabei werden Darstellungstechniken für zweidimensionale Medien oft nicht berücksichtigt: *"Die Graphiker wissen schon lange, wie man bestimmte Information möglichst prägnant durch Schrift und Bild auf einer Fläche darstellt: die Information soll nicht zu dicht und ausgewogen über die Fläche verteilt sein, und inhaltlich zusammengehörende Aussagen sollen nebeneinander plziert werden."* (/Nievergelt83/, S. 30)

- daß Auf- und Abrollen des Bildschirmausschnitts und das Umblättern (vgl. Abschnitt 3.4) praktisch nicht verwendet werden. Ein gewisses "Umblättern" findet nur in der Form statt, daß der Benutzer bei Systemausgaben, die für einen Bildschirmausschnitt zu umfangreich sind, den "Seitenwechsel" initiiert. Ein "Zurückblättern" gibt es dabei allerdings oft nicht.
- daß oft keine Groß- und Kleinschreibung benutzt wird, obwohl sie leichter erkennbar ist als einfache Großschreibung;
- daß eine Strukturierung durch unterschiedliche Schriftarten und -größen nicht stattfindet;
- daß Verwirrung erzeugt wird durch Verwendung zu vieler oder falscher Farben und/oder Helligkeitsabstufungen;
- daß Belästigungen für den Benutzer durch zu viele blinkende Bereiche oder zu starke Kontraste entstehen.

Bewegung als Gestaltungsmittel bietet sich vor allem für Simulationen an - etwa um zeitliche Veränderungen darzustellen -, ist aber zum einen in der Programmierung aufwendig, zum anderen wird eine erhebliche Rechenleistung dafür benötigt. Bewegung wird deswegen in CUU-Programmen nur vereinzelt eingesetzt.

Am häufigsten wird für CUU-Programme zur Eingabe die alphanumerische Tastatur in Verbindung mit systembedingten Funktionstasten verwendet, vereinzelt findet man auch allgemeine Funktionstasten und Steuerknüppel. Generell zu bemängeln ist auch hier wieder die Konzeptlosigkeit, welche Eingaben sinnvoll auf welchen Geräten durchzuführen sind, und die oft umständliche Handhabung für den Benutzer: *"Schlimm steht es, wenn er (der Benutzer) einen Fehler (in einem Eingabetext) entdeckt. Viele Programme erlauben die Korrektur falscher, aber noch nicht abgeschlossener Befehle nicht und lassen dem Benutzer keine andere Wahl, als die falsche Eingabe abzubrechen und noch einmal von vorne zu beginnen."* (/Nievergelt83/, S. 31)

9.2. Eine Dialogrealisierung für gelegentliche Benutzer

Für gelegentliche Benutzer wurde zum Abschluß des Kapitels 7 innerhalb des allgemeinen Dialograhmens eine abstrakte Dialogstruktur in Form eines Wechsels der Formen Unterbrechung und Auswahlangebot vorgeschlagen, wodurch gleichermaßen eine Steuerbarkeit durch den Benutzer und eine Führung seitens des Systems - im Sinne von Benutzerfreundlichkeit ("Benutzerführung", vgl. /Schulze84/) - erreicht werden konnte. Kriterien wie leichte Erlernbarkeit, Selbsterklärungsfähigkeit und Verlässlichkeit wird weiter entsprochen durch Konkretisierungen unter Berücksichtigung von Piktogrammen, Fenstern und Dialogstufen.

Für einen Vorschlag zur Realisierung, der die Aspekte zur Benutzerfreundlichkeit aufgreift und weiter fortführt, gehen wir zunächst von einer Gerätekonstellation aus, die auf der Ausgabeseite einen hochauflösenden Farbbildschirm und einen grafikfähigen Drucker, auf der Eingabeseite eine Tastatur und eine Maus mit Funktionstaste vorsieht. Diese Geräte sind hier nur als Beispiele ausgewählt und durch andere ersetzbar. Wichtig ist die durch sie mögliche Arbeitsweise, d.h. die durchzuführenden abstrakten Funktionen (im Sinne von GKS, /GKS83/). Andere Geräte können an die Stelle der oben genannten treten. Insofern sind hier Geräte wie Digitalisierbrett oder Lichtgriffel genauso berücksichtigt wie zukünftige Entwicklungen, z.B. die Sprachein- und -ausgabe oder in die Tischplatte integrierte Bildschirme.

In einem normalen Systemzustand - bei der Arbeit mit einem Anwendungsprogramm - ist die Bildschirmfläche entsprechend der Dialogtypen in vier Bereiche unterteilt: Ein Fenster für den Kontrolldialog, eines für den Gestaltungsdialog, eines für Metafunktionen und eines - das größte - für den Anwendungsdialog. Der Bildschirm könnte z.B. so organisiert sein, daß am oberen Rand ein schmaler Streifen vorgesehen wird, der je ein Feld für Kontroll-, Gestaltungs- und Metafunktionen beinhaltet. Der übrige Bereich dient dem Anwendungsdialog, wobei Fensterüberlappungen möglich sind. Eine unterschiedliche Darstellung durch verschiedene Farben für die Bereiche/Felder erscheint sinnvoll. Benz, Grob, Haubner /Benz81/ halten 9-11 Farbtöne für vom Menschen bei der Bildschirmaarbeit unterscheidbar und empfehlen eine Abstufung auf 6 bis 8 Farbtöne zur Codierung von Information. Berücksichtigt man farbige Darstellungen in Anwendungen, so sollte der Dialograhmen diese Zahl nicht voll ausschöpfen.

Eine farbliche Unterscheidung des Anwendungsdialogs einerseits (z.B. schwarze Schrift auf weißem Untergrund) und des Kontroll-, Gestaltungs- und Metadialogs (z.B. dunkelgraue/schwarze Schrift auf hellgrauem Untergrund) andererseits reicht evtl. zum Hervorheben unterschiedlicher Dialogtypen aus und verhindert Verwirrung durch zu viele Reizstufen. Für endgültige Aussagen zur farblichen Gestaltung des Bildschirms mit dem Ziel der Unterscheidung von Dialogtypen sind empirische Untersuchungen notwendig. Möglich ist auch, die Farbwahl vom System her offenzulassen und über die Dialogstufen, bzw. Gestaltungsfunktionen dem Benutzer die Wahl zwischen Grauton- und Mehrfarbendarstellung zu überlassen. Wichtig, aus physiologischen Gründen, ist bei der Wahl verschiedener Farben, daß keine zu großen Unterschiede im Helligkeitsgrad auftreten; die Technik des "inverse video", bei der Teile des Bildschirms statt normal (helle Schrift auf dunklem Untergrund), invertiert geschaltet werden (dunkle Schrift auf hellem Untergrund), ist deshalb abzulehnen (vgl. zur Begründung /Nievergelt83/ und /Benz81/).

Bereits angesprochen worden ist die Definition "typenzugehöriger" Cursor: bewegt man mittels der Maus den Cursor auf dem Bildschirm, so kann er im

Anwendungsdialog in Form eines kleinen senkrechten Strichs (bei textuellen Anwendungen) oder eines kleinen Fadenkreuzes (bei grafischen Anwendungen) dargestellt sein. Im Bereich von Kontroll-, Gestaltungs- oder Metadialog kann der Cursor in Form unterschiedlich gestalteter Pfeile realisiert werden. Wird in diesem Zusammenhang Texteingabe nötig, ist wieder der Übergang zu einem anderen Symbol ratsam. Hier kann, wie im Anwendungsdialog, ein kleiner senkrechter Strich verwendet werden (Realisierung des Apple Macintosh). Ebenso möglich ist ein kleiner waagerechter Strich, der unter der augenblicklichen Schreibposition erscheint (Realisierungsform des Cursors vieler Mikrorechner). Zeitspannen, in denen der Benutzer keine Eingaben tätigen kann, weil das System interne Aktionen zur Bearbeitung einer Benutzereingabe durchführen muß, können dem Benutzer ebenfalls durch die Form des Cursors angezeigt werden. Das Macintosh-System verwendet hier das Symbol einer Uhr, das Smalltalk-System das Symbol einer Brille. Sowohl Farbe als auch Form sorgen dann dafür, daß dem Benutzer der jeweilige Systemzustand deutlich wird.

Die Felder am oberen Bildschirmrand enthalten je ein Piktogramm für die dort aktivierbaren Funktionentypen. Gut geeignet, weil international bekannt und mit für diesen Zweck geeigneten Assoziationen verbunden, sind hierfür Piktogramme in Form von Wegweisern:

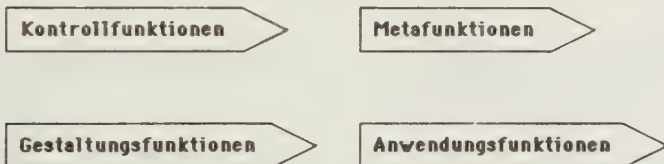


Abb. 9.1.: Piktogramme für Dialogtypen

Bezüglich der farblichen Gestaltung ist am oben Gesagten anzuknüpfen und eine Grautondarstellung einer mehrfarbigen (gelb/schwarz, blau/weiß) standardmäßig vorzuziehen. Zur Anpassung an unterschiedliche Benutzerwünsche erscheint im Zuge verschiedener Dialogstufen eine Umschaltung von Farb- und Grautondarstellung wünschenswert. Die in der abstrakten Dialogstruktur für den gelegentlichen Benutzer vorgesehene Arbeitsweise wird realisiert durch Bewegen des Cursors auf das gewünschte Piktogramm z.B. mithilfe der Maus (der Cursor ändert dabei seine Darstellung) und Auslösen der Maus-Taste. Hierdurch wird das System veranlaßt, die aktuelle Befehlsumgebung zu verlassen und die in der Zielklasse möglichen primären Funktionen als Menü zu präsentieren ("Pull-down-Menü" oder auch "Pop-up-Menü"). Aus diesem Menü wählt der Benutzer wieder durch Bewegen der Maus und damit des Cursors und der Auslösetaste die gewünschte Alternative aus. Dies setzt voraus, daß das präsentierte Menü entsprechend den möglichen Funktionen wiederum in Felder aufgeteilt ist.

Zur Darstellung der Funktionenfelder können - z.B. wieder mittels Dialogstufen umschaltbar - textuelle Form oder auch Piktogramme gewählt werden. Als Hintergrundfarbe ist der für diese Bereiche vorgesehene Grau- bzw. Farbton zu übernehmen.

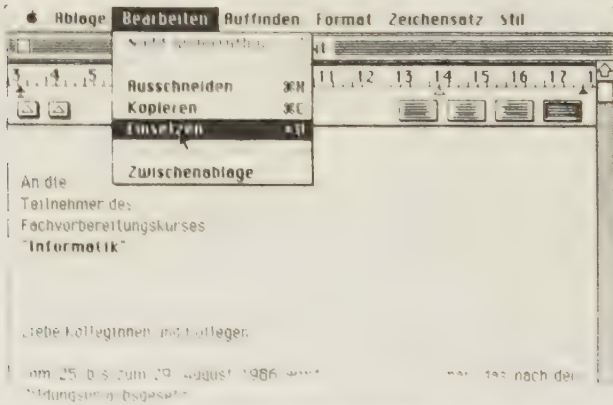


Abb. 9.2: Pull-down-Menü des Apple-Macintosh (Patentiert durch Apple)

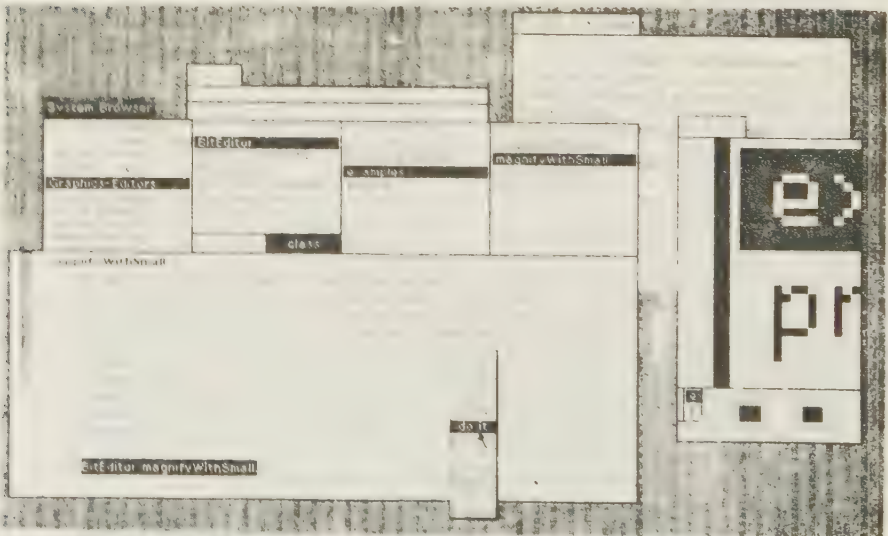


Abb. 9.3: Pop-up-Menü und Fenster in Smalltalk

Greift man die Aufteilung der Typen in primäre und sekundäre Funktionen wieder auf, so sollten grundsätzlich die möglichen primären Kontroll-, Gestaltungs- und Metafunktionen im Menü direkt ansprechbar sein. Die sekundären Kontroll-, Gestaltungs- und Metafunktionen kann man erreichen, indem jeweils ein Menüfeld - z.B. wieder durch das Piktogramm eines Wegweisers realisiert - als "Kopf" eines die sekundären Funktionen beinhaltenden Menüs fungiert.

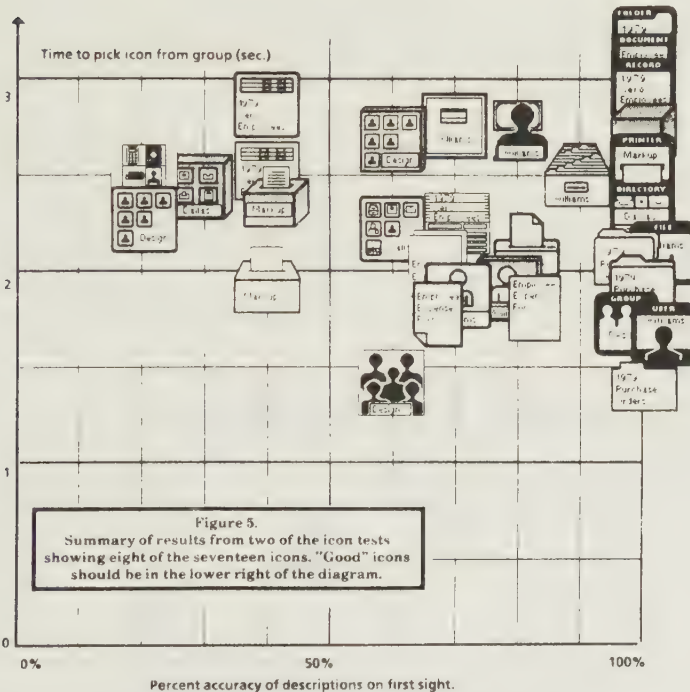


Abb. 9.4: "Gute" und "weniger gute" Piktogramme für den Bürobereich.
Aus: /Bewley83/.

Zur Darstellung der Menüalternativen in Piktogramm-Form kann mit der obigen Begründung die Anlehnung an Verkehrszeichen fortgeführt werden. Die jahrelang beim Benutzer gewachsenen Assoziationen zwischen Formen und Bedeutungen muß dabei ausgenutzt werden, indem sinnentsprechende Zuordnungen vorgenommen werden. Hierdurch gefördert werden z.B. Selbsterklärungsfähigkeit, Verlässlichkeit und leichte Erlernbarkeit des System-

verhaltens. Ein allzu strenges Festhalten an Verkehrszeichen ist wegen der in vielen Fällen fehlenden Entsprechung zwischen Bedeutung einer Operation innerhalb des Dialogsystems und der Bedeutung von Verkehrszeichen zu vermeiden. Hier sind Sinnbilder aus anderen Bereichen des täglichen Lebens (z.B. Büro) oft geeigneter. Genaue Aufschlüsse über die Eignung von Symbolen hinsichtlich bestimmter Aktionen sind u.a. auch aus empirischen Untersuchungen ableitbar. Stauer /Stauer85/ und Bewley /Bewley83/ beschreiben detailliert die Aspekte, die bei der Konstruktion von Piktogrammen zu beachten sind, und führen für den Bürobereich eine solche empirische Untersuchung durch (vgl. Abb. 9.4). Für die uns interessierende Unterrichtsumgebung fehlen solche Untersuchungen. Die Verwandtschaft dieses Bereichs zum Bürobereich läßt jedoch vermuten, daß die Ergebnisse von Stauer in vielen Fällen auf CUU-Systeme übertragbar sind.

Funktionen, die sich auf Fenster und Darstellungsfelder beziehen, sind als Piktogramme - der Macintosh benutzt dazu zwei Streifen am unteren und seitlichen Rand - in Form von Pfeilen und anderen Zeichen den Darstellungsfeldern direkt zuzuordnen. Eine Bewegung von Fenstern und von Darstellungsfeldern wird dann durch "Bewegung mit der Hand" durchgeführt. So wird eine Entsprechung zwischen physischer Benutzeraktion und Wirkung auf das System hergestellt ("Direct manipulation", "Direkte Manipulation").

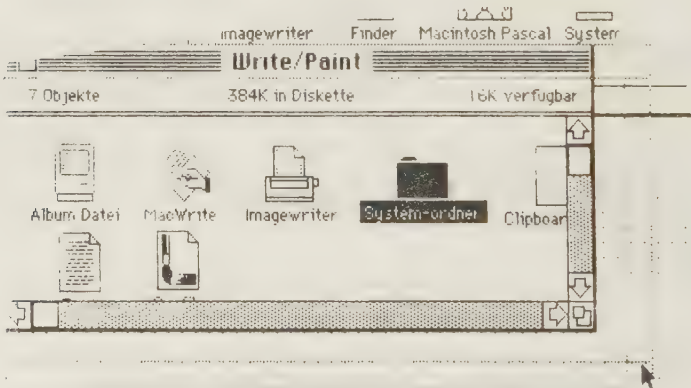


Abb. 9.5: Beispiel für eine direkte Manipulation beim Macintosh-System:
Vergrößern eines Fensters

Bei Text-Darstellung von Menüalternativen muß als Grundeinstellung der Text mit dem für geübte Benutzer vorgesehenen entsprechenden Kommandonamen übereinstimmen. Dies gewährleistet leichte Erlernbarkeit und auch Verläß-

lichkeit. Die Vorgehensweise des Apple-Macintosh, neben einen Text das entsprechende abgekürzte Kommando darzustellen, unterstützt das Lernen des Benutzers, im Umgang mit dem System vom gelegentlichen zum geübten Benutzer "aufzusteigen". Die Wahl der Kommando-Kürzel des Macintosh bleibt dem deutsch-sprachigen Benutzer rätselhaft, weil trotz sonst deutschen Bezeichnungen hier Kürzel von englischen Ausdrücken verwendet werden (vgl. Abb. 9.2), ein mit Sicherheit falsches Vorgehen.

Nach Auslösen einer Funktion wird in der Regel ein (rechnergeführter) Dialog zur näheren Spezifizierung der Funktion nötig. Dieser Dialog findet statt innerhalb eines zu diesem Zweck auf dem Bildschirm dargestellten Fensters/Darstellungsfeldes. Hintergrund- und Textfarbe sind hierfür entsprechend dem Dialogtyp zu wählen. Das Fenster ist weiter grafisch so zu gestalten, daß durch Auswahl einzelner Felder mittels der Maus und eventuell Felder zur Protokollierung von einzugebenden Texten (mittels der Tastatur) der Benutzer die notwendigen Parameter besetzt und das so gefüllte Fenster an das System überträgt - z.B. durch Auslösen eines Fertig-Feldes. Auf diese Weise kann der Benutzer Eingaben überprüfen, verändern, korrigieren und hat das Gefühl, selbst den Dialog zu führen.

Eine Unterbrechung dieses Dialogs ist dann auf die gleiche Weise durch Maus und Auslösetaste möglich. Das dann nicht mehr aktive Fenster wird optisch entsprechend gekennzeichnet, kann das vorhergehende (teilweise) überlagern oder auch neben dem alten auf dem Bildschirm untergebracht werden. Ein neues Fenster wird als aktives Fenster hervorgehoben. Ist der zwischen-geschobene Dialog beendet, verschwindet das entsprechende Fenster und das zuvor deaktivierte Fenster wird wieder als aktiver Arbeitsbereich ausgezeichnet. Daneben ist das Fenster- und Darstellungsfeld-Konzept für den Benutzer flexibel zu gestalten. D.h. der Benutzer kann "unten liegende" Darstellungsfelder "nach oben" bewegen oder auch mehrere nebeneinander auf dem Bildschirm plazieren. Innerhalb welcher Arbeitsumgebung er gerade kommuniziert, entscheidet er selbst durch Aktivieren des entsprechenden Fensters. Diese Arbeitsweise stellt eine Zusammenfassung der Funktionen "Verlasse aktuelle Arbeitsumgebung" und "Eröffne eine Anwendung" dar, bei der eine Zielanwendung direkt, ohne den "Umweg" über das Pop-up-Menü des Kontrolldialogs angegeben wird. Es sollten beide Vorgehensweisen dem gelegentlichen Benutzer parallel zur Verfügung stehen.

9.3. Eine Dialogrealisierung für geübte Benutzer

Auf der Grundlage der zuletzt vorausgesetzten Gerätekonstellation wird der geübte Benutzer im Verhältnis zum gelegentlichen kaum mit Hilfe der Maus, sondern stattdessen mehr mit der Tastatur arbeiten. Die in einem Systemzustand

möglichen Funktionen sind ihm bekannt, in Zweifelsfällen informiert er sich mit Hilfe von Metafunktionen.

Von diesen Voraussetzungen ausgehend, ist der gesamte Bildschirm dem Anwendungsdialog vorbehalten. Darstellungsfelder für Kontroll-, Gestaltungs- und Metafunktionen werden nur bei Bedarf abgebildet. Eine Ausnahme bilden hier wieder die auf Fenster und Darstellungsfelder bezogenen Funktionen. Sie sind wiederum als Piktogramme den Bildern zugeordnet und werden durch Maus- und Auslösetaste angesprochen. Bezüglich der farblichen Gestaltung gelten die von den gelegentlichen Benutzer angeführten Bemerkungen (vgl. den letzten Abschnitt).

Einen Anwendungsdialog kann der Benutzer durch Eingabe eines Kommandos zur Aktivierung einer gewünschten Funktion einer der anderen Typen verlassen. Einzuweisen ist dieses Kommando durch ein den Typ spezifizierendes Steuerzeichen. Auf dem Bildschirm wird dann in einem die Anwendung überlagernden Darstellungsfeld die Eingabe protokolliert. Der Benutzer hat dabei jederzeit die Möglichkeit, etwaige Tippfehler zu beseitigen. Durch ein Abschlußsymbol wird die Eingabe als beendet gekennzeichnet und kann vom Rechner verarbeitet werden. Etwa benötigte Parameter, Namen und Gerätespezifikationen können bei der Eingabe des Kommandos direkt eingegeben werden oder innerhalb des Darstellungsfeldes durch einen der Kommandoingabe folgenden Dialog geklärt werden. Auch hier wird durch Farbe, Cursor und grafische Aufbereitung des Darstellungsfeldes dem Benutzer die augenblickliche Arbeitsumgebung verdeutlicht.

Ein erneutes Unterbrechen durch Eingabe eines Kommandos führt zur Überlagerung des nicht mehr aktiven Fensters durch ein neues, das erst nach vollständiger Bearbeitung der Funktion wieder vom Bildschirm verschwindet. Bezüglich der Handhabung von Fenstern gilt weiter das für den gelegentlichen Benutzer Gesagte: Neben der direkten Kommandoingabe über Tastatur kann der geübte Benutzer auch durch die Maustechnik einzelne Funktionen aktivieren. Streng genommen handelt es sich dann zwar um einen Menüdialog - der Benutzer wählt zum "Eröffnen einer Anwendung" aus den auf dem Bildschirm in Darstellungsfeldern dargebotenen Anwendungen aus -, doch wird die wachsende Verantwortung nicht als Nachteil empfunden. Im Gegenteil ist diese Art Wechsel von Anwendungen mit geringerem Aufwand verbunden als z.B. die Kommandoingabe über Tastatur.

4.4. Zur Einbettung anwendungsorientierter Funktionen

Im anwendungsorientierte Kontroll- und Gestaltungsfunktionen in den allgemeinen Dialograhmen benutzerfreundlich zu integrieren, müssen sie gegenüber gelegentlichen Benutzern in die entsprechenden Pop-up-Menüs,

gegenüber geübten Benutzern in den Kommandovorrat aufgenommen werden. Ihre Handhabung unterscheidet sich nicht von anderen Kontroll- und Gestaltungsfunktionen, d.h., für sie gelten die Bemerkungen in den letzten beiden Abschnitten entsprechend.

Anwendungsfunktionen sollten ebenfalls zur Gewährleistung einheitlicher Arbeitsweise auf die beschriebene Weise realisiert werden. Deshalb müssen die Funktionen dieses Dialogtyps für gelegentliche Benutzer in einem Darstellungsfeld - z.B. wieder durch Pop-up-Menüs - angezeigt werden, um dem Benutzer die Orientierung im System zu ermöglichen. Eine Aktivierung der Funktionen erfolgt dann durch Dialogschritte entsprechend der Arbeitsweise von gelegentlichen Benutzern. Diese Forderungen für die Gestaltung von Anwendungsprogrammen gewährleistet bestmögliche Integration von Anwendungen in den allgemeinen Dialograhmen und führt zu einem insgesamt benutzerfreundlichen Verhalten von CUU-Systemen.

Für geübte Benutzer sind Anwendungsfunktionen als Kommandos zu realisieren, wobei wieder ein Fenster zur Klärung von Parametern oder auch eine direkte Angabe von Parameterwerten vorzusehen ist. Ein Steuerzeichen kennzeichnet das Kommando als zum Anwendungsdialog gehörig. Bezüglich der möglichen Anwendungsprogramme wollen wir in diesem Zusammenhang drei Gruppen unterscheiden.

Die erste Gruppe beinhaltet die durch den Dialograhmen als Autorensystem speziell für den CUU erstellten Programme, wie z.B. Drill & Practice, Tutorielle Programme, Simulationen, Pascal-Interpreter usw. Dazu gehören aber auch schulspezifische Informationssysteme. Programme dieser Gruppe sind innerhalb der schuleigenen Mikrorechnernetze oder auf dem Schulrechner, der zweiten Hierarchieebene der in Kapitel 6 vorgestellten Architektur, installiert; ihre Benutzerschnittstelle entspricht der Arbeitsweise des allgemeinen Dialograhmens.

Die zweite Gruppe von Anwendungsprogrammen bilden Programme, die standardmäßig in der CUU-Umgebung notwendig sind. Hier wäre als Beispiel die Taschenrechner-Funktion zu nennen, die es dem Benutzer jederzeit ermöglicht, Berechnungen zwischendurch durchzuführen. Diese Anwendung muß als Anwendungsfunktion Bestandteil jeder anderen Anwendung sein und ist insofern Teil des Dialograhmens. Z.B. XEROX-Star und Apple-Macintosh realisieren diese Anwendung durch einen auf dem Bildschirm darstellbaren Taschenrechner, dessen "Tastatur" mithilfe der Maus oder der Computertastatur bedient wird. In entsprechender Weise kann solch ein Taschenrechner als stets verfügbare Anwendungsfunktion - initialisierbar durch Auslösen eines dafür auf dem Bildschirm im Anwendungsberereich vorgesehenen Feldes (Piktogramm) - in den Dialograhmen integriert werden. Eine entsprechende Vorgehensweise ist für andere dieser Gruppe zuzurechnenden Anwendungen (z.B. ein Duden, ein Lexikon

oder ein Wörterbuch) zu wählen. Ihre Erstellung ist dann Teil des Dialograhmens.

Die dritte Gruppe umfaßt alle Anwendungen, die als Dialograhmen-unabhängig anzusehen sind. Sie sind auf allen Hierarchieebenen möglich und sind charakterisiert durch eine von dem allgemeinen Dialograhmen abweichende Schnittstelle. Für sie sind Auswertungsprogramme nötig, um sie in die Gruppe 1 zu überführen.

9.5. Zwischenformen bei der Realisierung

Die im zweiten und dritten Abschnitt dieses Kapitels vorgestellten Realisierungen bezogen sich auf den gelegentlichen und den geübten Benutzer und stellen damit zwei Extreme dar, die in dieser Form nicht allen Benutzertypen gerecht werden. Der geübtere gelegentliche Benutzer wird durch die eine Realisierung unter-, der weniger geübte Benutzer durch die andere überfordert. Weiter ist zu beachten, daß ein Benutzer gegenüber einer Anwendung als geübter, gegenüber einer anderen als weniger geübter Benutzer eingestuft werden kann.

Zwischenformen durch Dialogstufen, mit denen die Ausführlichkeit von Systemausgaben, Farbgebungen, Piktogrammen und textuellen Darstellungen beeinflusbar sind, wurden bereits angesprochen, ebenso die Möglichkeit, durch Gestaltungsfunktionen Regeln für Benutzereingaben zu formulieren. Eine weitere Zwischenform entsteht durch die Definition einer Dialogstufe, die die äußere Präsentation der Realisierung für gelegentliche Benutzer vorsieht, alternativ aber auch eine Kommandoingabe über Tastatur akzeptiert. Der Benutzer kann so bei ihm bekannten Funktionen einen Kommandodialog führen, bei ihm unbekannten den Unterbrechungs-Menu-Dialog mit Hilfe der Maus.

Diese Form wird vor allem denjenigen Benutzern gerecht, die gegenüber dem allgemeinen Dialograhmen als geübte Benutzer einzustufen sind, denen aber einzelne benötigte Anwendungen fremd sind, sich also hier wie gelegentliche Benutzer verhalten.

10. Dialoggrundfunktionen für den allgemeinen Dialograhmen

Das Konzept des allgemeinen Dialograhmens wurde unter anderem durch die Notwendigkeit geräte- und anwendungsunabhängiger Dialoggrundfunktionen begründet (vgl. Kapitel 6). Solche Dialoggrundfunktionen können einerseits benutzerorientiert und andererseits geräteorientiert diskutiert werden. Beide Aspekte werden im Bereich der grafischen Datenverarbeitung berücksichtigt. Wir wollen im folgenden die dort gefundenen Funktionen erörtern und allgemein auf interaktive Systeme übertragen.

10.1. Geräteorientierte Dialoggrundfunktionen

Die zuletzt vorgestellte Arbeitsweise, mittels Maus und Auslösetaste aus einer Menge auf dem Bildschirm dargestellter Funktionen eine gewünschte auszuwählen, läßt sich z.B. ebenso realisieren durch

- Verwendung eines Lichtgriffels oder des Fingers, indem direkt auf das auf dem Bildschirm dargestellte Element gewiesen wird, oder durch
- einen Digitizer, für den die Menge der Funktionen auf einer Schablone eingetragen ist und durch den Digitalisierstift eine Funktion ausgelöst wird, oder auch durch
- Funktionstasten, die in die Tastatur integriert sind.

Wichtig ist hier also nicht, durch welches Gerät eine Funktion ausgelöst wird, sondern welche abstrakte Funktion durch das Gerät realisiert ist. Im oben beschriebenen Fall handelt es sich um das Auswählen einer Funktion aus einer vorgegebenen Alternativmenge. In der grafischen Datenverarbeitung werden solche abstrakten Funktionen durch die Norm GKS /DIN 66252/ festgelegt. Für den Bereich der Eingabe sind definiert:

Pick	Identifizieren eines dargestellten Objekts;
Locator	Auswahl eines Punktes auf einem Ausgabegerät;
Choice	Auswahl einer Möglichkeit aus einer vorgegebenen Alternativmenge;
Valuator	Auswahl eines Zahlenwertes aus einem vorgegebenen Intervall;
Stroke	Angabe einer Punktfolge;
String	Eingabe einer Zeichenkette.

Sie sind innerhalb dreier unterschiedlicher Eingabemodi verwendbar: **Request, Sample, Event** (vgl. /DIN 66252/). Als abstrakte Ausgabefunktionen sind vorgesehen:

Polyline	Ausgabe eines Kantenzuges;
Polymarker	Ausgabe einer Punktmenge in Form von Punktsymbolen;

Text	Ausgabe einer Zeichenkette;
Fill Area	Ausgabe einer Fläche; *
Cell Array	Ausgabe eines Rasterbildes;
Generalized Drawing Primitive	Ausgabe spezieller grafischer Elemente (z.B. Kurven).

Man kann diese Funktionen ohne Einschränkungen auf den CUU-Bereich, aber auch allgemein auf interaktive Systeme übertragen. Mit Hilfe des in GKS vorgesehenen Fenster- und Arbeitsplatzkonzeptes, sowie der den Ausgabeobjekten zuzuordnenden Attribute (vgl. /DIN 66252/) lassen sich durch diese Funktionen sämtliche für den Dialograhmen geforderten Realisierungen nachbilden. Als Beispiele seien genannt:

- einzelne Darstellungsbereiche für Dialogtypen werden durch Definition von Fenstern und Darstellungsfeldern erreicht,
- Textein- und -ausgaben durch die Funktionen STRING und TEXT
- Menü-Eingaben durch CHOICE,
- Aktivieren und Unterbrechen von Anwendungen u.a. durch die Funktion PICK (Identifizieren eines Fensters),
- Bewegen von Objekten (z.B. Verschieben von Fenstern) durch CHOICE und LOCATOR oder STROKE.

Verschiedene Darstellungen von Objekten (Texten, Piktogrammen, Fenstern) erreicht man durch das Setzen von Attributen (Linientyp, Farbindex, Schriftart, Schriftgröße usw.)

Die Aufnahme der geräteorientierten Dialoggrundfunktionen sorgt für Geräteunabhängigkeit des allgemeinen Dialograhmens. Bezüglich der Gestaltung des Dialogverhaltens und des Übergangs zu Anwendungsprogrammen bieten diese Funktionen allerdings keine ausreichende Unterstützung. Hierzu sind Dialogfunktionen aus Benutzersicht zu definieren.

10.2. Benutzerorientierte Dialoggrundfunktionen

Für den Bereich grafische Datenverarbeitung liegt eine Untersuchung von Foley [Foley84] vor, in der die für grafische Ein- und Ausgaben möglichen logischen Funktionen aus der Sicht eines Benutzers diskutiert werden. Die Bezeichnung "logische Funktionen" wird hier von uns zur Unterscheidung von den mehr geräteorientierten "abstrakten Funktionen" gewählt. Logische Funktionen sind Grundfunktionen für den Benutzer zur Durchführung einer bestimmten Aktion innerhalb eines Mensch-Rechner-Dialogs; sie stehen nicht in Zusammenhang mit den "logischen Funktionen" der mathematischen Logik oder den "boolean functions" in Programmiersprachen. Nach Foley sind sechs logische Eingabefunktionen zu unterscheiden:

Select	Der Benutzer wählt eine Möglichkeit aus einer Menge von Alternativen aus.
Position	Der Benutzer wählt einen Punkt des Bildschirms aus, um dort ein Objekt zu plazieren.
Orient	Der Benutzer gibt einem dargestellten Objekt eine neue Orientierung, er dreht es.
Path	Der Benutzer gibt eine Folge von Bildschirmpunkten bzw. Orientierungen an und beschreibt damit eine zeitliche Eigenschaft von Objekten.
Quantify	Der Benutzer gibt einen Zahlenwert ein.
Text	Der Benutzer gibt eine Zeichenfolge ein.

Eine Übertragung dieser benutzerorientierten logischen Funktionen als ebensolche innerhalb des für den allgemeinen Dialograhmen relevanten Bereichs ist möglich, allerdings erscheint diese Aufteilung nicht zwingend. Innerhalb des Kontroll-, Gestaltungs- und Metadialogs wurde die Select- (bei gelegentlichen) oder die Text-Funktion (bei geübten Benutzern) verwendet, um eine neue Funktion auszulösen. Die darauffolgenden Dialogschritte zur Parameterfestsetzung bestanden wieder aus Select- und Text-Operationen, aber auch Werteingabe (Quantify) wurde hier nötig, z.B. zur Einstellung der Tastaturempfindlichkeit, des Kontrastes usw. Anwendungsdialoge beinhalten Grafik- (z.B. bei Simulationen) und Texteingaben. Mit der im letzten Kapitel beschriebenen Arbeitsweise lassen sich auch hier Eingaben im wesentlichen auf Select, Text und Quantify zurückführen, bei grafischen Anwendungen finden zusätzlich Position, Orient und Path Verwendung (z.B. bei Programmen zum Erstellen von Zeichnungen).

In der Aufzählung von Foley fehlt die Funktion Pick (Identifizieren eines Objekts), die in der grafischen Datenverarbeitung als elementar angesehen wird und durch GKS genormt ist (s.o.). Stattdessen führt er vier "Kontrollfunktionen" ein (Stretch, Sketch, Manipulate, Shape), die aber mehr als Mischfunktionen für Ein- und Ausgaben zu interpretieren sind und die zum Teil die Pick-Funktion benötigen. Unter der Annahme, daß das Identifizieren eines Objekts immer mit der Absicht verbunden ist, dieses Objekt irgendwie in seiner Darstellung zu verändern, erscheint aus Benutzersicht diese Vorgehensweise als möglich. Allerdings sollte Orient in diese Gruppe übernommen werden.

Insgesamt wirkt Foleys Unterteilung zwar teilweise unsystematisch, aber er nennt Grundtätigkeiten im Umgang mit grafischen Systemen und kann auf diese Weise Aussagen über benutzerfreundliche Realisierungstechniken mit verfügbaren Geräten treffen (vgl. /Foley84/), - seine Ergebnisse stimmen mit den Aussagen des letzten Kapitels überein. Wir wollen uns für das Folgende nicht zwingend auf die von Foley gefundenen Funktionen als Grundfunktionen zur Gewährleistung von Anwendungsunabhängigkeit festlegen, sondern setzen ohne nähere Untersuchung über Art und Zahl sogenannte logische Funktionen als Bindeglied

zwischen den geräteunabhängigen abstrakten Funktionen und anwendungsorientierten Funktionen voraus. Sicherlich können die von Foley beschriebenen Funktionen ein erster Ansatz solcher logischer Funktionen sein.

Die Benutzerschnittstelle, d.h. der allgemeine Dialograhmen legt für einen Mensch-Rechner-Dialog ein Benutzermodell zugrunde (gelegentlicher, geübter Benutzer, Zwischenformen). Das Anwendungsprogramm wendet sich durch eine anwendungsorientierte Funktion an den allgemeinen Dialograhmen, um seine Aktion eines Dialogschritts zum Benutzer zu senden. Der allgemeine Dialograhmen überführt diese zunächst in logische Funktionen und realisiert entsprechend dem Benutzermodell die Gestaltung des Dialogs. Danach findet dann eine Transformation in die abstrakten Dialoggrundfunktionen statt, um die eigentliche Ausgabe geräteunabhängig vornehmen zu können.

Die Aktion des Benutzers wird entsprechend entgegengesetzt dem Anwendungsprogramm zur Verfügung gestellt. Eine Eingabe wird zunächst durch eine GKS-Funktion dargestellt und dann - mithilfe des Benutzermodells - vom Dialograhmen zunächst in logische und dann in anwendungsabhängige Form überführt und an das Anwendungsprogramm übergeben.

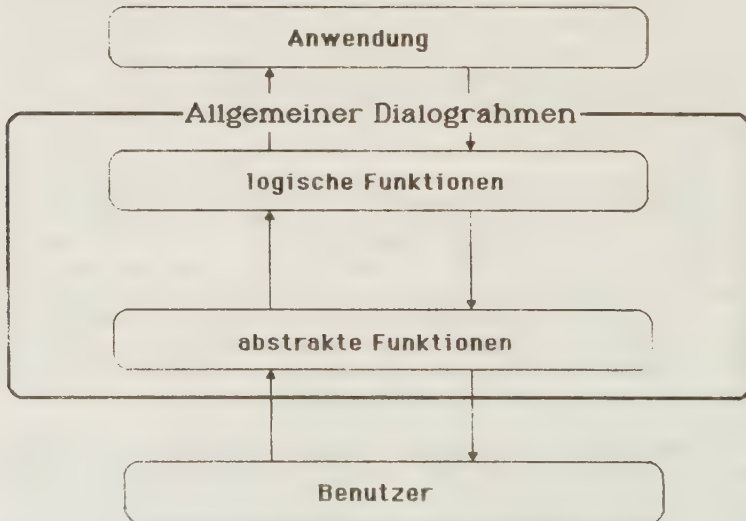


Abb. 10.1. Transformationen im allgemeinen Dialograhmen

Teil 3: Modellierung

Durch die Diskussion der Möglichkeiten und Anforderungen bei Mensch-Rechner-Dialogen wurde mit dem allgemeinen Dialograhmen ein System verbal beschrieben, das zur Bildung benutzerfreundlicher Mensch-Rechner-Dialoge (nicht nur für den Bereich CUU) in die Praxis umgesetzt, d.h. programmiert werden muß. Mit Hilfe der Implementierung kann dann eine Erprobung der verschiedenen Konzepte zur Gestaltung des Mensch-Rechner-Dialogs erfolgen.

Wir wollen in diesem dritten Teil der Arbeit Modelle zur Beschreibung von interaktiven Systemen herleiten, die - angewendet auf den allgemeinen Dialograhmen - zum einen das dynamische Verhalten während eines Mensch-Rechner-Dialogs wiedergeben und zum anderen formal genug sind, um aus ihnen direkt die Programmierung abzuleiten.

11. Ein Modellierungskonzept für Mensch-Rechner-Dialoge

Nachdem in den vorangegangenen Kapiteln ausführlich begründet und beschrieben wurde, **was** an Systemeigenschaften zu fordern ist für zukünftige benutzerfreundliche interaktive Computersysteme, soll in den folgenden Kapiteln dargestellt werden, **wie** diese Forderungen in die Praxis umgesetzt werden können. Dabei werden allerdings die Programmierung und Implementierung nicht direkt betrachtet, sondern die Funktionsweise des allgemeinen Dialograhmens wird mit Hilfe formaler Modelle beschrieben. Diese Modelle bilden die notwendige Voraussetzung für eine Implementierung und sind logisch auf der **Anwendungsschicht des ISO-Referenzmodells** (vgl. z.B. /Diebold86/) anzusiedeln.

Zur modellhaften Darstellung des dynamischen Verhaltens von Mensch-Rechner-Dialogen eignen sich prinzipiell eine Reihe unterschiedlicher Beschreibungsverfahren diskreter dynamischer Systeme ("Kommunikationsgraphen", "Petri-Netze", "markierte Petri-Netze", "Evaluationsnetze", "State Diagrams"). Saathoff /Saathoff82/ vergleicht einige dieser Verfahren und findet für ein dem allgemeinen Dialograhmen vergleichbares Problem markierte Petri-Netze als geeignetstes Mittel zur Darstellung formaler Modelle für den Mensch-Rechner-Dialog.

Wir wollen uns für das Folgende ihren Begründungen und Ausführungen anschließen und eine spezielle Form von markierten Petri-Netzen, die - von uns so bezeichneten - **strukturierten, markierten Netze** zur Bildung formaler Modelle des allgemeinen Dialograhmens verwenden; unter anderem auch deshalb, um Teile ihrer Ergebnisse hier übernehmen und weiterführen zu können.

Eine Einführung und Begriffsbildung zur Verwendung markierter Petri-Netze in Zusammenhang mit Mensch-Rechner-Dialogen gibt Oberquelle /Oberquelle76/. Einige wesentliche Grundzüge seiner Arbeit stellen wir im ersten Abschnitt dieses Kapitels kurz vor und entwickeln daraus im zweiten Abschnitt ein für unser Problem geeignetes Modellierungskonzept. Dieses Instrumentarium wird dann im dritten Abschnitt dazu benutzt, in die Methode einzuführen, indem ein einfacher CUU-Dialog durch Petri-Netze modelliert wird. Eine Ausweitung auf den allgemeinen Dialograhmen erfolgt dann in den anschließenden Kapiteln.

11.1. Petri-Netze

Definition

Ein **Petri-Netz (Netz)** N ist eine mathematische Struktur, bestehend aus zwei nichtleeren, endlichen, disjunkten Mengen K und I und einer nichtleeren Relation R , die nur Elemente verschiedener Mengen

miteinander verknüpft und kein Element aus K und I isoliert:

$N = (K, I, R)$ mit

(i) K und I sind disjunkte Mengen und

(ii) $R \subset (K \times I) \cup (I \times K)$ ist eine zweistellige

Relation, die kein Element aus K oder I isoliert
ist ein Petri-Netz.

Elemente aus K heißen **Kanäle**, Elemente aus I **Instanzen**. Ein Petri-Netz wird auf der Grundlage dieser Begriffe oft auch **Netz aus Instanzen und Kanälen** (kurz: **NIK**) genannt.

(Andere oft verwendete Begriffe für Kanäle sind Stellen oder Bedingungen, für Instanzen Transitionen oder Ereignisse, vgl. /Reisig82/.)

Grafisch lassen sich NIKs durch Verwendung von Kreisen für Kanäle, Rechtecken für Instanzen und Pfeilen für Elemente der Relation R darstellen.

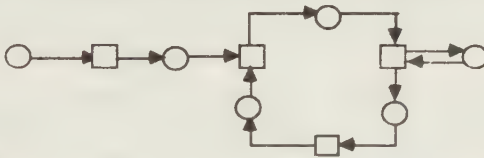


Abb. 11.1 : Beispiel eines Netzes aus Instanzen und Kanälen

Für eine Anwendung von Petri-Netzen auf Mensch-Rechner-Dialoge interpretiert man Instanzen als "Informationsverarbeiter" (Prozessoren) und Kanäle als "Informationsträger". Für Kanäle ist darüberhinaus eine Unterteilung sinnvoll in solche, die Information zum Zwecke der Speicherung tragen (**Zustandskanal**), und in solche, die Information zum Zwecke der Übertragung, Weiterleitung tragen (**Übergabekanal**). Des weiteren erweisen sich die Begriffe Vorbereich und Nachbereich einer Instanz oder eines Kanals als nützlich:

Definition Sei $N = (K, I, R)$ ein Petri-Netz. Für $x \in K \times I$ heißt
 $V_x := \{ y \mid (y, x) \in R \}$ der **Vorbereich** und
 $x_N := \{ y \mid (x, y) \in R \}$ der **Nachbereich** von x

Mit dieser Interpretation, bei der - etwas gewöhnungsbedürftig, weil das Spektrum menschlicher Fähigkeiten auf die Funktionen "Gedächtnis" und "Verarbeiten" abgebildet wird - der Mensch und der Rechner durch Instanzen und Kanäle dargestellt werden, kann der statische Aspekt, das ist die Konfiguration des "Mensch-Rechner-Systems", die für einen Dialog notwendig ist, wie in Abb. 11.2. modelliert werden.

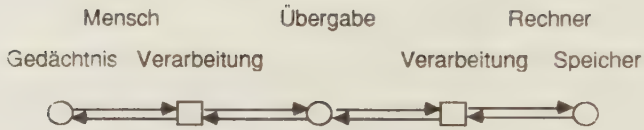


Abb. 11.2.: Einfaches Modell für den Mensch-Rechner Dialog

Oberquelle nennt diese Ebene der Abstraktion die **Organisationsebene** und definiert ein **dialogfähiges System** durch Verfeinerung von Abb. 11.2., indem er beim Übergabekanal die Funktionen Benutzerein- und Systemausgabe trennt (Abb.11.3.). Diese Aufteilung ist auf den ersten Blick zwar plausibel, bei näherer Betrachtung heutiger Dialogsysteme allerdings nicht ausreichend: Mensch und Rechner kommunizieren nur über einen Ein- und einen Ausgabekanal. Das Vorhandensein mehrerer Ein- und Ausgabegeräte wird hier nicht dargestellt. Wie dies durch Verfeinerung möglich ist, beschreiben wir ausführlich in Abschnitt 11.3.1.

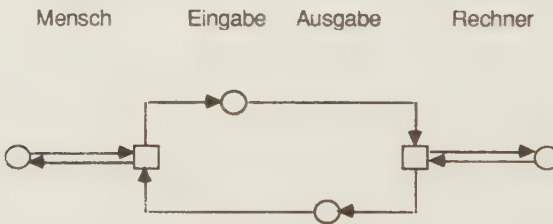


Abb.11.3.: Dialogfähiges System auf der Organisationsebene

Neben der Organisationsebene führt Oberquelle zwei Abstraktionsebenen zur Modellierung des Mensch-Rechner-Dialogs an, mit denen auch eine Darstellung eines konkreten Dialogs (**Prozeßebe**ne, vgl. Abb. 11.4.) zwischen dem Menschen und dem Rechner und eine Darstellung von Dialogklassen (**Systemebene**, vgl. Abb. 11.5.) möglich werden. Auf der Prozeßebe

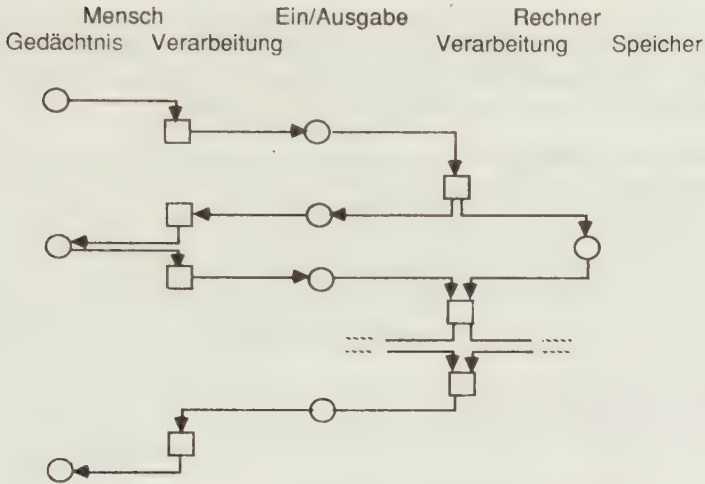


Abb. 11.4.: NIK auf Prozessebene mit zeitlicher Abfolge von Dialogschritten (nach /Oberquelle76/, S. 18)

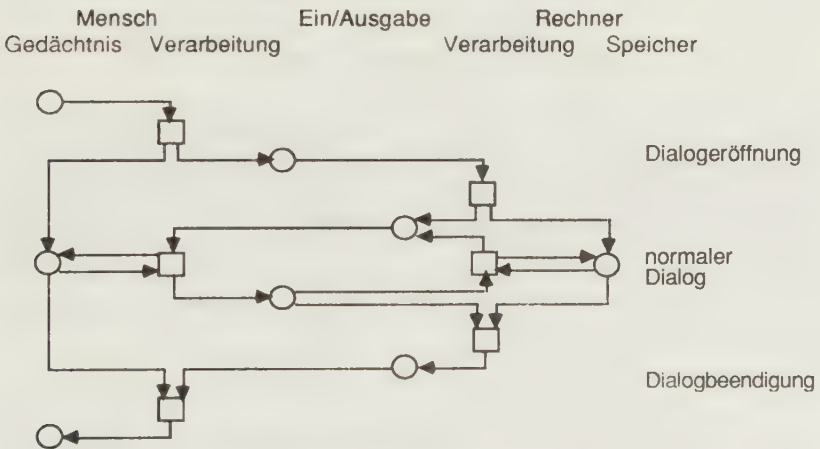


Abb. 11.5.: NIK auf Systemebene mit den Dialogklassen "Dialogeröffnung", "normaler Dialog" und "Dialogbeendigung" (nach /Oberquelle76/, S. 16)

Ausgangspunkt für diese beiden Ebenen ist, daß der Mensch mit dem Dialog beginnt - er agiert - und der Rechner auf Aktionen des Menschen reagiert. Oberquelle /Oberquelle76/ verwendet aus diesem Grund die Namen "Dialog-

f"hrer" f"ur die "menschliche Instanz" und "Dialognehmer" f"ur den Rechner; Bezeichnungen, die wir wegen der Problematik von "Dialogf"hrung" (vgl. Kapitel 3 und 6) nicht "ubernehmen wollen. Wohl aber wird weiterhin davon ausgegangen, da" der Mensch von sich aus den Dialog startet.

Ferner definiert Oberquelle ein umfangreiches Begriffssystem f"ur Mensch-Rechner-Dialoge (vgl. /Oberquelle76/, S. 23-41), das bis heute allerdings nicht allgemein akzeptiert ist und in den weiteren Betrachtungen f"ur uns auch nicht von Bedeutung ist. Zwei Begriffe (Zustandskanal und "Ubergabekanal") wurden bereits genannt, zwei weitere werden im folgenden benutzt: Als **Eingangskanal** einer Instanz wird derjenige Kanal bezeichnet, von dem aus eine Verbindung zu der Instanz f"uhrt. Entsprechend ist derjenige Kanal, der durch eine Verbindung von einer Instanz aus erreicht wird, ein **Ausgangskanal**.

Eingangskanal Instanz Ausgangskanal



Abb. 11.6.: Ein- und Ausgangskanal einer Instanz

Definition

Sei $N = (K, I, R)$ ein Netz. F"ur $x \in I$ hei"t
 $e \in K$ **Eingangskanal** von x , falls $e \in V_x$ gilt und
 $a \in K$ **Ausgangskanal** von x , falls $a \in x_N$ gilt.

Noch nicht ausreichend durch die bisherigen Modelle erfa"t ist das dynamische Verhalten des "Mensch-Rechner-Systems". Zwar lassen sich bei der Modellierung auf Proze"ebene Dialogschrittfolgen nachvollziehen und bei Modellierung auf Systemebene durch wiederholtes Durchlaufen von Netzteilen eine gewisse Dynamik nachbilden. Die Auswirkungen von Dialogschritten auf Speicherinhalte, die Auswirkungen von Verarbeitungen in Instanzen auf Inhalte von "Ubergabekan"len, das Verh"ltnis von "Ubergabe einer Nachricht und Reaktion auf diese Nachricht ist aber nicht ausreichend modelliert.

11.2. Markierte Petri-Netze

Um eine bessere Anpassung des Modellierungskonzepts an die Bedingungen innerhalb von Mensch-Rechner-Dialogen zu erreichen, ist ein "Ubergang zu markierten Petri-Netzen n"tig.

Definition

Ein **markiertes Petri-Netz** (**markiertes Netz**) ist ein Petri-Netz, erweitert um
 - zwei Abbildungen (der Markierung und der Kapazit"t), die den Kan"len eines Petri-Netzes

- natürliche Zahlen zuordnen und
- eine Abbildung G (das Gewicht), die den Relationen eines Petri-Netzes natürliche Zahlen zuordnet.

$M_N = (K, I, R, C, M, G)$ mit

- (i) (K, I, R) ist ein Netz,
- (ii) $C : K \rightarrow \mathbb{N}$ erklärt eine Kapazität für jeden Kanal,
- (iii) $M : K \rightarrow \mathbb{N}$ ist eine Markierung für jeden Kanal, die die Kapazität respektiert, d.h. für jeden Kanal $x \in K$ gilt: $M(x) \leq C(x)$ und
- (iiii) $G : R \rightarrow \mathbb{N}$ bestimmt zu jedem Element aus R ein Gewicht

ist ein markiertes Petri-Netz.

Grafisch dargestellt wird ein markiertes Petri-Netz im allgemeinen durch Einzeichnen von Punkten (den Marken) in die die Kanäle darstellenden Kreise (Abb.11.7.); weiter werden die Kreise mit den Kapazitäten und die Pfeile mit den Gewichten beschriftet.

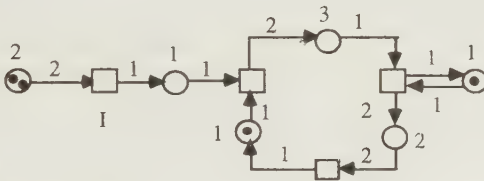


Abb. 11.7.: Ein markiertes Petri-Netz

Das dynamische Verhalten von Mensch-Rechner-Dialogen wird nun durch markierte Petri-Netze nachgebildet, indem, ausgehend von einer **Anfangsmarkierung**, durch das **Schalten einer Instanz** eine **Nachfolgemarkierung** erzeugt wird.

Definition

Sei $M_N = (K, I, R, C, M, G)$ ein markiertes Petri-Netz. Die Instanz $x \in I$ **schaltet von einer Markierung M zu einer Markierung M'** $\Leftrightarrow \forall y \in K$ gilt:

$$M'(y) := \begin{cases} M(y) - G(y, x), & \text{falls } y \in V_x \setminus x_N, \\ M(y) + G(x, y), & \text{falls } y \in x_N \setminus V_x, \\ M(y) - G(y, x) + G(x, y), & \text{falls } y \in x_N \cap V_x \\ M(y) & \text{sonst.} \end{cases}$$

Die Instanz $x \in I$ heißt **schaltfähig**, falls

- (i) $\forall y \in V_x$ gilt: $M(y) \geq G(y, x)$ und
- (ii) $\forall y \in x_N$ gilt: $C(y) \geq M(y) + G(x, y)$.

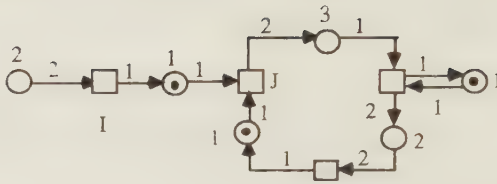


Abb. 11.8 Markiertes Petrinetz nach Schalten von I (Die Instanz J ist jetzt schaltfähig)

Abb. 11.8. zeigt das markierte Petri-Netz, das durch Schalten der Instanz I aus dem aus Abb. 11.7. hervorgeht. Will man nun einzelne Dialogschritte durch solche markierten Petri-Netze nachbilden, so müssen die Marken stellvertretend für alle in dem modellierten Mensch-Rechner-Dialog zur Verfügung stehenden Funktionen stehen. Hierzu sind die Marken dann jeweils entsprechend zu interpretieren, was bei einem umfangreichen Funktionsvorrat wiederum ein unbefriedigender Zustand ist. Zwar lassen sich durch Markierungen und Folgemarkierungen Dialogschritte und -zustände prinzipiell nachvollziehen und unterscheiden, inhaltliche Aspekte von Dialogschritten und -zuständen werden aber nicht nachgebildet: Hat der Benutzer eine Datenabfrage beantwortet, ein Kommando eingegeben oder den Rechner unterbrochen? Wie ändert sich die System-Datenbasis bei einer Benutzeraktion? Saathoff [Saathoff82] führt deshalb Instanzenvorschriften und Attribute für Marken zur Detaillierung der Modelle ein.

Instanzenvorschriften sind Algorithmen, die als **Verarbeitungsvorschriften** - Saathoff benutzt hier die Bezeichnung "einfache Instanzenvorschriften" - die speziellen Aufgaben von Instanzen beschreiben, d.h. in welcher Form eine Instanz auf eine Nachricht reagiert, und die als **Konfliktlösungsvorschriften** die Aufgabenverteilung unter verschiedenen Instanzen regeln, wenn beim Schalten ein Kanal Eingangskanal mehrerer Instanzen ist und entschieden werden muß, welche von ihnen schalten sollen. **Attributsmarken** sind als Objekte der Instanzenvorschriften zu interpretieren. Der jeweilige Algorithmus nimmt eine Attributsmarke auf, wertet sie aus, führt entsprechende Verarbeitungsschritte durch und erzeugt eine Attributsmarke für seine Ausgangskanäle. Eine Unterscheidung von **Übergabemarken**, die die durch das System fließenden Nachrichten beschreiben, und **Zustandsmarken** zur Darstellung von Systemzuständen auf Zustandskanälen bietet sich an (vgl. [Saathoff82], S.158). Darüberhinaus ist eine Strukturierung der Attribute entsprechend den Funktionen eines Dialogsystems notwendig, um im konkreten Fall Inhalte von Dialogschritten und die Wirkung von Instanzenvorschriften darstellen zu können. Wie solch eine Strukturierung vorzunehmen ist, wird für einen einfachen CUU-Dialog unten einleitend beschrieben und dann systematisch für den

allgemeinen Dialograhmen in Kapitel 14 ausgeführt. Zuvor jedoch noch einige Bemerkungen zu dem sich aus den letzten Überlegungen ergebenden Modellierungskonzept.

11.3. Strukturierte, markierte Netze

Bei Berücksichtigung von Instanzenvorschriften und strukturierten Attributsmarken im Modell in der oben beschriebenen Weise verändern wir das zugrundeliegende Modellierungskonzept der markierten Petri-Netze in wesentlichen Teilen. So spielt das Gewicht von Elementen der Relation R beim Schalten keine Rolle mehr: Wie aus einer Ausgangsmarkierung durch das Schalten einer Instanz eine Nachfolgemarkierung erzeugt wird, legen in erster Linie die Instanzenvorschriften fest. Weiter ist die Interpretation einer Markierung als Zuordnung einer natürlichen Zahl zu einem Kanal mit Festlegung einer Kapazität als Obergrenze überflüssig, wenn wir strukturierte Attributsmarken als wesentliche Beschreibung für die Kanäle vorsehen.

Wir definieren deshalb sogenannte **strukturierte, markierte Netze** zur Modellierung von Mensch-Rechner-Dialogen und benutzen dazu die Begriffe Datenstruktur und Algorithmus.

Definition Eine **Datenstruktur** wird verstanden als Zusammenfassung von Objekttypen nach ihrem Zweck, ihrer Bedeutung oder ihrer Bearbeitungsweise. Eine **Ausprägung** einer Datenstruktur liegt vor, wenn die der Struktur zugeordneten Objekttypen mit konkreten Werten belegt sind.

Als **Algorithmus** bezeichnen wir hier eine Abbildung, die in der Menge der Datenstrukturen operiert und auf der Grundlage von Ausprägungen von Datenstrukturen neue Ausprägungen von Datenstrukturen erzeugt.

Sei D eine Menge von Datenstrukturen, A_D eine Menge von Ausprägungen von Datenstrukturen und A eine Menge von Algorithmen, so ist $a \in A$ zu beschreiben durch

$$a : A_D \times \dots \times A_D \rightarrow A_D \times \dots \times A_D.$$

Sei weiter K_N die Menge aller Nachbereiche aller Kanäle $x \in K$ eines Netzes. Damit ist dann

$$S_M_N = (K, I, R, S, SA, IV, IK)$$

(i) (K, I, R) ist ein Netz,

(ii) $S : K \rightarrow \{0, 1\}$ ist eine Markierung für jeden Kanal zur Beschreibung einer Situation,

- (iii) $SA : K \rightarrow D$ ist eine Zuordnung einer Datenstruktur (strukturierte Attributsmarke) zu jedem Kanal,
- (iiii) $IV : I \rightarrow A$ ist eine Zuordnung eines Algorithmus (der Verarbeitungsvorschrift) zu jeder Instanz,
- (iiiii) $IK : K_N \rightarrow A$ ist eine Zuordnung eines Algorithmus (der Konfliktlösungsvorschrift) zu jedem Nachbereich von Kanälen aus K

ein strukturiertes, markiertes Netz.

Die Abbildung S ordnet dabei jedem Kanal (wie in markierten Petri-Netzen die Abbildung M) eine natürliche Zahl zu, um das dynamische Verhalten von Mensch-Rechner-Dialogen nachbilden zu können. Wir beschränken uns hier allerdings auf die Zahlen 0 und 1 als mögliche Werte der Abbildung und zeigen dadurch die augenblickliche Situation des Mensch-Rechner-Dialogs an. Eine Zuordnung der "1" zu Kanälen zeigt an, daß die entsprechenden Datenstrukturen mit aktuellen Werten belegt sind - hier also Ausprägungen vorliegen - und die Instanzen der Nachbereiche schalten, d.h. die Objekte verarbeiten können (sofern sie schaltfähig sind). Zustandskanäle besitzen während des gesamten Dialogs die Situationsmarkierung "1". Schaltfähigkeit und das Schalten von Instanzen sind - abweichend von den Festlegungen des letzten Abschnitts - jetzt folgendermaßen zu definieren:

Definition

Eine Instanz $x \in I$ eines strukturierten, markierten Netzes $S_M_N = (K, I, R, S, SA, IV, IK)$ heißt **schaltfähig**
 $\Leftrightarrow \forall v \in V_x$ gilt $S(v) = 1$.

Die Instanz $x \in I$ eines strukturierten, markierten Netzes $S_M_N = (K, I, R, S, SA, IV, IK)$ **schaltet**

- (i) von einer Situations-Markierung S zu einer neuen Situations-Markierung S'

$\Leftrightarrow \forall y \in K$ gilt:

$$S'(y) := \begin{cases} 0 & , \text{ falls } y \in V_x \setminus x_N \\ 1 & , \text{ falls } y \in x_N \setminus V_x, \\ 1 & , \text{ falls } y \in x_N \cap V_x, \\ S(y) & , \text{ sonst.} \end{cases}$$

- (ii) von einer Ausprägung einer strukturierten Attributsmarkierung A_D_{SA} zu einer neuen Ausprägung einer strukturierten Attributsmarkierung $A_D'_{SA}$

$\Leftrightarrow \forall y \in K$ gilt:

$$A_D'SA(y) := \begin{cases} \text{undef.}, & \text{falls } y \in V_x \setminus x_N, \\ (IV(x))(A_DSA(y_1') \times \dots \times (A_DSA(y_{N'})) & \\ \quad \text{mit } y_1', \dots, y_{N'} \in V_x, & \\ \quad \text{falls } y \in x_N \setminus V_x & \\ (IV(x))(A_DSA(y_1') \times \dots \times (A_DSA(y_{N'})) & \\ \quad \text{mit } y_1', \dots, y_{N'} \in V_x, & \\ \quad \text{falls } y \in x_N \cap V_x, & \\ A_DSA(y) & \text{sonst.} \end{cases}$$

Die Definition besagt, daß das Schalten einer Instanz jetzt zweierlei bewirkt: Zum einen wird durch die Situations-Markierung S' eine neue Dialog-Situation angezeigt, also der dynamische Prozeß des Mensch-Rechner-Dialogs nachgebildet. Zum anderen wird der Algorithmus der schaltenden Instanz auf die Ausprägungen der Datenstrukturen der Eingangskanäle angewendet und dadurch neue Ausprägungen der Datenstrukturen der Kanäle des Nachbereichs der Instanz erzeugt, bzw. bei Zustandskanälen verändert. In Abbildungen von strukturierten, markierten Netzen zeichnen wir einen Punkt in die die Kanäle darstellenden Kreise zur Nachbildung der Dialogsituation S , wenn $S(y) = 1$ für $y \in K$. Die Datenstrukturen und Algorithmen führen wir gesondert auf.

Mit dem so definierten Modellierungskonzept haben wir ein Werkzeug geschaffen, das einerseits den dynamischen Prozeß und andererseits auch inhaltliche Aspekte von Mensch-Rechner-Dialogen nachbilden kann. Darüberhinaus bilden die Algorithmen für Instanzenvorschriften und die Datenstrukturen für strukturierte Attributsmarken einen ersten Ansatz zur Realisierung des modellierten Systems.

11.4. Modelle für einen einfachen CUU-Dialog

Ausgangspunkt für die folgenden Betrachtungen ist das Tutorielle Programm "Deutsche Grammatik, 2. Zeichensetzung" (/Intus84/, S.8). Dessen Dialogverhalten ist typisch für einfache CUU-Dialoge und soll deshalb hier durch Modelle dargestellt werden. Das Programm ist auf Diskette gespeichert und z.B. auf Apple II-Rechnern lauffähig.

11.4.1. Organisationsmodelle eines einfachen CUU-Dialogs

Auf der Organisationsebene ergibt sich als Modell das Bild eines einfachen dialogfähigen Systems mit Diskette/Arbeitsspeicher als Zustandskanal des Rechners, der Tastatur als Eingabekanal und Bildschirm, sowie Tongenerator als Ausgabekanäle (vgl. Abb. 11.9).

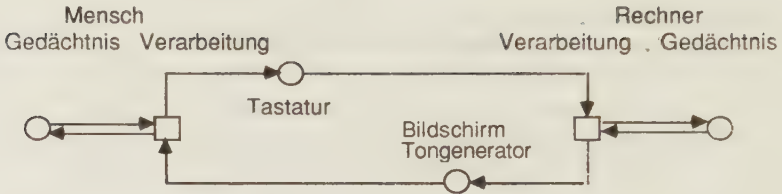


Abb. 11.9. Organisationsmodell für das Programm "Zeichensetzung"

Wie in Abschnitt 11.1 angemerkt, modelliert dieses Netz die Ein- und Ausgabe über mehrere Geräte nur ungenügend. Die naheliegendste Lösung, je Gerät einen eigenen Kanal vorzusehen (Abb. 11.10.), erweist sich wegen der Schaltregel beim Übergang zu markierten Petri-Netzen ebenfalls als problematisch. Die "menschliche Instanz" kann nur schalten, wenn beide

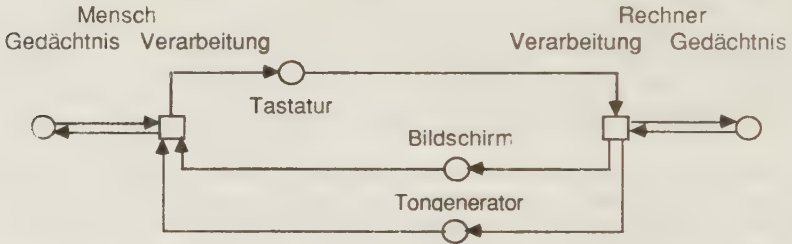
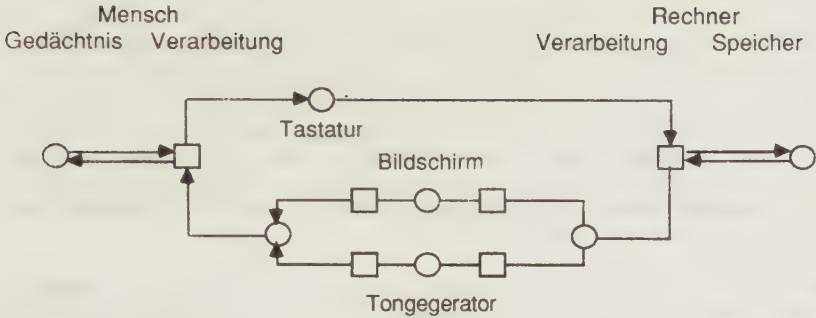


Abb. 11.10.: Organisationsmodell für das Programm Zeichensetzung mit zwei Ausgabekanälen

Eingabekanäle markiert sind, also über beide Geräte gleichzeitig eine Information ausgegeben wird. Dies ist zwar speziell bei diesem Programm der Fall, allerdings kann man nicht allgemein in Dialogsystemen solch eine Praxis voraussetzen. Ein drittes Modell für die Organisationsebene ergibt sich durch Verfeinerung des Netzes aus Abb. 11.9., bei der jedes E/A-Gerät durch eine Instanz-Kanal-Kombination nachgebildet wird (Abb. 11.11.).

Interpretiert man in Abb. 11.11. die Ausgabekomponenten als "Geräteprozessor", "Ausgabemedium" und aufnehmendes Sinnesorgan" und berücksichtigt einseitig einen "Ausgabesteuerungskanal" und am Ende einen "Übertragungs- und Koordinationskanal", so erhält man ein realistischeres Modell für eine Konfiguration mit mehreren Ausgabegeräten. Für die Eingabeseite ist bei Bedarf entsprechend zu verfahren.

Damit hier gleichzeitige Ausgaben über mehrere Geräte nachempfunden werden können, ist die Konfliktlösungsvorschrift im Anschluß an den "Ausgabesteuerungskanal" entsprechend zu formulieren (siehe Abschn. 11.4.2.)



Abt. 11.11. Organisationsmodell für das Programm Zeichensetzung mit Detaillierung der Ausgabe

Die Koordination der Eindrücke der "aufnehmenden Sinnesorgane" im Falle mehrerer aktiver Ausgabe-Geräte ist allerdings vom Modell her wiederum ein Problem, weil hierzu eine Markierung für den "Übertragungs- und Koordinationskanal" durch mehrere Instanzen gleichzeitig aufgebaut wird. Hier dürfen die Instanzen nicht einfach nur ihre Marken weiterleiten, sondern es ist eine Synthese der Arbeit mehrerer Instanzen notwendig. Diesen, im Kopf des Menschen stattfindenden Vorgang lassen wir im folgenden unberücksichtigt.

Es sei zum Schluß noch darauf hingewiesen, daß Saathoff /Saathoff82/ von Modellen der Gestalt aus Abb. 11.9 ausgeht und diese so interpretiert werden, daß nur das gerade aktive Gerät durch den Kanal dargestellt wird, mehrere gleichzeitig arbeitende Ausgabegeräte also nicht vorgesehen sind. Weiter ist auch denkbar, daß das Netz aus Abb. 11.10. verwendet und in der Form interpretiert wird, daß tatsächlich jedes Gerät immer zur Ein- oder Ausgabe dient. Die laut Programm nicht aktiven Geräte tragen dann die Information, daß sie keine Information tragen. Wir wollen im folgenden solche Interpretationen nicht zugrundelegen, sondern gehen von der in Abb. 11.11. beschriebenen Konstellation aus.

11.4.2. Systemmodell eines einfachen CUU-Dialogs

Vor der Darstellung von Modellen auf Systemebene zunächst einige verbale Erläuterungen zum Ablauf des betrachteten tutoriellen Programms:

Nach Einlegen der Diskette mit dem Lernprogramm und Einschalten des Gerätes erzeugt das System ein Titelbild und fragt nach dem Namen des Benutzers mit

der Aufforderung, die Eingabe (über Tastatur) durch eine "Eingabe-Taste" abzuschließen. In Folge werden mehrere "Bildschirmseiten" mit textlichen Erklärungen ausgegeben, wobei der Wechsel von einer Seite zur nächsten durch die "Eingabe-Taste" vom Benutzer veranlaßt wird. Auf ein Beispiel muß der Benutzer durch Eingabe von "a" oder "b" eine Entscheidung treffen, die der Rechner durch Text und akustische Ausgabe kommentiert, bevor für die weitere Arbeit vom Programm ein Menü ausgegeben wird. Mithilfe dieses Menüs hat der Benutzer die Möglichkeit, unterschiedliche Programmteile zum Erlernen der verschiedenen Zeichensetzungsregeln (z.B. "Kommata in Aufzählungen") auszuwählen, bzw. die Arbeit abubrechen.

Nach Wahl einer Alternative (Eingabe einer Ziffer) werden wieder mehrere "Bildschirmseiten" Erklärungen ausgegeben, bevor Testfragen vom Benutzer beantwortet werden müssen. Jede Testfrage wird durch eine Erklärung (inverse video, vgl. Abschnitt 9.2) eingeleitet. Die Testfrage selbst besteht in der Ausgabe eines Satzes, der mit Satzzeichen zu ergänzen ist. Dazu wird der Cursor vom Programm auf Positionen innerhalb des Satzes bewegt, an denen der Benutzer dann das Satzzeichen ergänzt oder durch die "Eingabe-Taste" entscheidet, kein Satzzeichen zu setzen. Fehler des Benutzers werden durch eine akustische Ausgabe (Piepen) für jede falsche Entscheidung quittiert. Nach Durcharbeiten des Satzes erfolgt im Falle von vorangegangenen Fehlern eine Ausgabe von auf den Testsatz bezogenen Interpunktionsregeln.

Auf die Bearbeitung mehrerer Testsätze folgt die Ausgabe, wieviele Sätze nicht korrekt bearbeitet wurden und dann wieder das Ausgangsmenü mit dem der Benutzer über seine weitere Arbeit entscheiden kann.

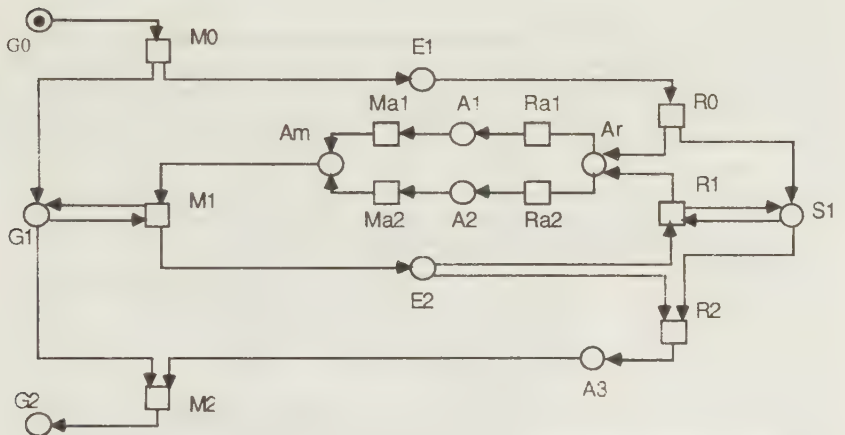
Zur Modellierung des Programmablaufs auf Systemebene ist von dem strukturierten, markierten Netz aus Abb. 11.12. auszugehen.

Daneben ist die Festlegung von Instanzenvorschriften und Attributsmarken durchzuführen. Generell soll durch diese Festlegungen eine Darstellung der Kausalität einzelner Dialogschritte erreicht werden, wobei vom deterministischen Verhalten der Rechnerkomponenten ausgegangen werden muß.

Die "menschlichen Komponenten" des Modells weisen ein solches Verhalten nicht auf. Der Mensch wählt aus Funktionen eine gewünschte aus, wobei das Verfahren - die Denkweise, die zu einer Auswahl führt - nicht durch Algorithmen beschrieben werden soll. Aus diesem Grund erfolgt die Festlegung der Zustandskanal-Marken für G0, G1 und G2, sowie der Verarbeitungsvorschriften für M0, M1 und M2 informell. Allerdings wird soetwas wie "die gute Absicht" des Menschen dem Modell zugrundegelegt. D.h. die Verarbeitungsvorschriften für R0, R1 und R2 setzen sinnvolle Eingaben durch den Menschen voraus und können dann den Dialog nachbilden. In der "Realität" ignoriert das System "falsche" Eingaben in der Form,

- daß zum Fortsetzen von Ausgaben nicht nur die "CR-Taste", sondern bis auf spezielle Funktionstasten (RESET, ESCAPE, CONTROL) alle Tasten der Tastatur als "Eingabe-Taste" interpretiert werden,
- daß für die Menüauswahl solange eine Eingabe gefordert wird, bis die ausgelöste Taste einen sinnvollen Wert hat (zwischen 1 und 8),
- daß bei Testsätzen jede Eingabe außer der korrekten als falsch gewertet wird (z.B., wenn der Benutzer statt eines Kommas einen Buchstaben oder ein Semikolon eingibt).

Die folgenden Festlegungen orientieren sich an dieser Vorgehensweise.



Legende:

G0, G1, G2:	Gedächtnis innerhalb der drei Dialogklassen (vgl. Abb.11.5)
M0, M1, M2:	"menschlicher Prozessor" innerhalb der drei Dialogklassen
E1, E2:	Eingabekanäle zur Dialogöffnung und im normalen Dialog
A1, A2:	Ausgabekanäle für Bildschirm und Tongenerator im normalen Dialog
A3:	Ausgabekanal bei Dialogbeendigung
Ma1, Ma2:	"Sinnesorgane"
Ra1, Ra2:	Geräteprozessoren
Am:	Übertragungs- und Koordinationskanal
Ar:	Ausgabesteuerungskanal
R0, R1, R2:	Rechner-Instanz innerhalb der drei Dialogklassen
S1:	Speicher

Abb. 11.12.. Systemmodell für das Programm Zeichensetzung

Marken auf Übergabekanälen

Zur Wiedergabe von Aktion und Reaktion der Dialogschritte eines Mensch-Rechner-Dialogs strukturiert Saathoff /Saathoff82/ die Attributsmarken entsprechend den in einem System möglichen Funktionen; und benutzt diese dann auch zur Konfliktlösung zwischen Instanzen, bzw. als Grundlage der Verarbeitungsvorschrift einer Instanz. Sie erhält durch die Festlegung, daß alle Marken aller Übergabekanäle gleich strukturiert sind, sehr umfangreiche, teilweise unübersichtliche Marken, die sich zur Modellierung zwar prinzipiell als geeignet erweisen, jedoch bei mächtigen Dialogsystemen umständlich und unhandlich sind.

Wir wollen zur Modellierung konkreter Dialoge mit dem Zeichensetzungsprogramm wegen der geringen Zahl von Funktionen entsprechend vorgehen und analysieren dazu zunächst den Ablauf des Dialogs. Dieser findet in unterschiedlichen Phasen statt: Nach einer Einleitungsphase mit Titelbild und Nennung der Autorennamen folgt eine längere allgemeine Beschreibung, nach der der Benutzer aus sieben Lektionen eine auswählt. Entsprechend seiner Wahl erfolgt eine Phase von Erklärungen zur Lektion, bis dann der Benutzer Testsätze bearbeiten kann. Wir unterscheiden für unser Modell demzufolge die Phasen "Beschreibung", "Erklärung", "Testsätze" und sehen als Funktionen vor:

Dialogeröffnung	zum Starten des Dialogs durch den Benutzer; diese Funktion ist keiner der oben genannten Phasen zugeordnet und wird realisiert durch das Starten des Dialogprogramms (bei AppleII: Einlegen der Startdiskette).
Dialogende	zum Beenden oder Abbrechen des Dialogs durch den Benutzer; diese Funktion ist in allen Phasen auslösbar und wird durch eine spezielle Interrupt-Taste erreicht (bei AppleII: Ausschalten der Geräte);
Texteingabe	zum Eingeben eines Namens durch den Benutzer innerhalb der Beschreibungsphase;
Ausgabe_fortsetzen	als Benutzerfunktion, um nach einer Ausgabe eine weitere Systemausgabe zu veranlassen; dies wird realisiert durch Auslösen einer beliebigen Taste der Tastatur (keine Funktionstaste, s.o.) innerhalb der Phasen Beschreibung und Erklärung;
Ja_Nein	zur Beantwortung einer Frage innerhalb der Beschreibungsphase durch den Benutzer;
Menüauswahl	zur Auswahl einer von sieben Lektionen, bzw., zum Beenden des bestehenden Dialogs durch den Benutzer am Ende der Beschreibungsphase;
Zeichenauswahl	innerhalb der Phase der Testsätze als Benutzerfunktion zur Eingabe eines Satzzeichens, bzw. Entscheidung, daß kein Satzzeichen gesetzt werden soll;

Textausgabe	zur Ausgabe von textlichen Nachrichten durch das System auf dem Bildschirm in allen o.g. Phasen;
Inverse_Textausgabe	als Systemfunktion in der Phase Testsätze, um eine Textausgabe besonders hervorzuheben;
Tonausgabe	innerhalb der Phasen Beschreibung und Testsätze zur Ausgabe akustischer Signale durch das System;
Cursor	als Systemfunktion in der Phase Testsätze, um in einem Testsatz den Cursor auf die nächste Position einzustellen, an der ein Satzzeichen gesetzt werden kann.

Da während einer Aktion mehr als eine Funktion durchgeführt werden kann (z.B. Textausgabe und Tonausgabe ohne zwischenzeitliche Reaktion des Benutzers), wählen wir für unser Modell eine Funktionenmenge als eine Komponente der Übergabemarken. Die zweite Komponente bilden wir dann durch die zu übergebende Nachricht und sehen diese als einen Text an. In einer Pascal-ähnlichen Notation erhalten wir dadurch als Übergabemarke

```

Type      Text = Array[1..Maxint] of char;
Funktionen = (Dialogeröffnung, Dialogende, Texteingabe, Ausgabe_fortsetzen,
Ja_Nein, Menüauswahl, Satzzeichen_Entscheidung, Textausgabe,
Inverse_Textausgabe, Tonausgabe, Cursor);
Übergabemarke = Record
Funktion: Set of Funktionen;
Nachricht: Text
End

```

Marken auf Zustandskanälen

Auch hier wählt Saathoff /Saathoff82/ eine Unterteilung in einzelne Funktionen ("Eingabe zu XY erwartet", "Dialog eröffnet") zur Strukturierung von Marken, führt dies allerdings nur ansatzweise für ein einfaches Beispiel aus. Wir wollen für die Zustandsmarke S1 entsprechend vorgehen (s.u.). Die "menschlichen Zustandskanäle" G0, G1 und G2 wollen wir nicht detailliert strukturieren, sondern sehen hier nur eine Marke vor, die den eröffneten, bzw. beendeten Dialog beschreibt:

```

Type Zustandsmarke_G0 = (Dialog_noch_nicht_eröffnet);
Zustandsmarke_G1 = (Dialog_eröffnet Dialog_beendet);
Zustandsmarke_G2 = (Dialog_ist_beendet)

```

Die Information über einen eröffneten oder beendeten Dialog wollen wir innerhalb unseres Modells auch für die Zustandsmarke auf S1 zur Verfügung stellen, um dadurch die Funktion der Instanzen R0 und R2 beschreiben zu können. Für eine Darstellung der Arbeit von R1 innerhalb des Dialogs sind weitere Attribute für die Zustandsmarke auf S1 notwendig. Diese ergeben sich aus dem dem Zeichensetzungsprogramm zugrundeliegenden Konzept der

unterschiedlichen Phasen (s.c.) und der Tatsache, daß das Programm durch den angeforderten Benutzernamen und durch die Registrierung von Zeichensetzungsfehlern ein Benutzerbild erstellt. Weiter wollen wir durch ein Attribut "Nachricht" semantische Eigenschaften von Dialogschritten nachbilden. Hier sollen Systemausgaben und -reaktionen dargestellt und für die Übergabe zum Benutzer bereitgehalten werden. Insgesamt erhält unsere Marke auf dem Zustandskanal S1 folgende Struktur:

```
Type Zustandsmarke_S1 = Record
    Nachricht: Text;
    Zustand: (Eröffnet, Beendet);
    Phase: (Beschreibung, Erklärung, Testsätze),
    Benutzer: Record
        Name: Text;
        Fehler, Richtig: Integer
    End
End
```

Verarbeitungsvorschriften

Die Verarbeitungsvorschriften stellen wir durch Prozeduren wieder in Pascal-ähnlicher Notation dar. Wie oben angesprochen gehen wir dabei für die "menschlichen Instanzen" informell vor und bilden nur für die Recheninstanzen das formale Verhalten als Algorithmus nach. Mit den Festlegungen für die Übergabe- und Zustandsmarken können wir die folgenden Prozeduren definieren:

```
Procedure Verarbeitungsvorschrift_M0;
Begin
    Zustandsmarke_G1 := Dialog_eröffnet;
    Übergabemarke.Funktion := [Dialogeröffnung];
    Übergabemarke.Nachricht := 'Dialogprogramm ist gestartet'
End;
```

```
Procedure Verarbeitungsvorschrift_R0;
Function NeueNachricht: Text; {...}
Begin
    Zustandsmarke_S1.Nachricht := NeueNachricht;
    Übergabemarke.Nachricht := Zustandsmarke_S1.Nachricht;
    Übergabemarke.Funktion := [Textausgabe];
    Zustandsmarke_S1.Zustand := Eröffnet;
    Zustandsmarke_S1.Phase := Beschreibung;
    Zustandsmarke_S1.Benutzer.Fehler := 0;
    Zustandsmarke_S1.Benutzer.Richtig := 0
End;
```

Wir haben hier eine Funktion NeueNachricht eingeführt, die entsprechend dem Dialogkontext die jeweilige Systemreaktion zur Verfügung stellen soll, und können dadurch inhaltliche Aspekte des Dialogs, d.h. welche Systemausgaben

zu einem Zeitpunkt tatsächlich durch das Zeichensetzungsprogramm erzeugt werden, auf einfache Weise in unser Modell integrieren.

Procedure Verarbeitungsvorschrift_Ra1;

Begin

{ Stelle die Nachricht des Systems auf dem Bildschirm dar und übergebe die unveränderte Übergabemarke an den Ausgangskanal }

End;

Procedure Verarbeitungsvorschrift_Ra2;

Begin

{ Stelle die Nachricht des Systems durch den Tongenerator dar und übergebe die unveränderte Übergabemarke an den Ausgangskanal }

End;

Procedure Verarbeitungsvorschrift_Ma1;

Begin

{ Nehme Bildschirminhalt auf und übergebe die unveränderte Übergabemarke an den Ausgangskanal }

End;

Procedure Verarbeitungsvorschrift_Ma2;

Begin

{ Nehme Signale des Tongenerators auf und gebe die unveränderte Übergabemarke an den Ausgangskanal }

End

Die Bedeutung der letztgenannten vier Instanzen ist zu Beginn dieses Abschnitts erläutert worden. In Bezug auf unser Modell besteht ihre Funktion lediglich in der Nachbildung unterschiedlicher Ausgabegeräte. Wie gerätetechnische Umformungen von Nachrichten erfolgen oder in welcher Form der Synthese-Prozeß im Gehirn des Menschen für die Verarbeitung von optischen und akustischen Signalen vor sich geht, bleibt für unser Modell unberücksichtigt.

Procedure Verarbeitungsvorschrift_M1;

Var Entscheidung: **Set of** Funktionen;

Entscheidungsinhalt: Text;

Begin {Verarbeitungsvorschrift_M1}

{ Analysiere die Nachricht und treffe sinnvolle Entscheidung für den nächsten Dialogschritt. Besetze Entscheidung und Entscheidungsinhalt mit Werten }

Case Entscheidung **of**

[Dialogende]

Begin

Übergabemarke.Funktion := [Dialogende];

Übergabemarke.Nachricht := 'Dialogprogramm ist beendet'

End;

[Texteingabe]

Begin

Übergabemarke.Funktion := [Texteingabe];

Übergabemarke.Nachricht := Entscheidungsinhalt

End;

[Ausgabe_fortsetzen] **Begin**

Übergabemarke.Funktion := [Ausgabe_fortsetzen];

```

Übergabemarke.Nachricht := Entscheidungsinhalt
End;
Begin
  Übergabemarke.Funktion := [Ja_Nein];
  If Entscheidungsinhalt = 'Wahl von a'
    Then Übergabemarke.Nachricht := 'a'
    Else Übergabemarke.Nachricht := 'b'
  End;
[Menüauswahl] Begin
  Übergabemarke.Funktion := [Menüauswahl];
  If Entscheidungsinhalt = 'Keine Lektion auswählen'
    Then Übergabemarke.Nachricht := 'Ende'
    Else Übergabemarke.Nachricht := Entscheidungsinhalt
  End;
[Zeichenauswahl] : Begin
  Übergabemarke.Funktion := [Zeichenauswahl];
  If Entscheidungsinhalt = 'Kein Satzzeichen'
    Then Übergabemarke.Nachricht := CR
    Else Übergabemarke.Nachricht := Entscheidungsinhalt
  End
End {case}
End {Verarbeitungsvorschrift_M1}

```

In der Prozedur wurde nicht erklärt, wie der Mensch zu einer Entscheidung kommt, sondern nur eingegrenzt, welche Entscheidungen vom Programmziel her sinnvoll und notwendig sind. Irrtümern bei einer Eingabe oder auch dem Versuch, eine "sinnlose Eingabe" zu tätigen, "um zu sehen, wie der Rechner reagiert", wurde hier begegnet, indem bei [Ja_Nein], [Auswahlangebot] und [Zeichenauswahl] die If-Abfrage so interpretiert wird, daß sie auf solche Eingaben den Wert TRUE liefert.

Zur Modellierung durch eine Prozedur *Verarbeitungsvorschrift_R* der rechnerseitig den Dialog abwickelnden Instanz R1 ist wieder eine Funktion *NeueNachricht* notwendig, um inhaltliche Aspekte des Dialogs nachzubilden. Zur besseren Verständlichkeit einiger Programmteile wurden mit der gleichen Bedeutung wie *NeueNachricht* weitere Funktionen definiert. Weiter benötigen wir einige Boole'sche Variablen, deren Sinn unmittelbar aus der Namensgebung ersichtlich ist, sowie eine Integer_Funktion "*Letzte_Cursor_Position*", die als Ergebnis die Cursor-Position innerhalb eines Textes liefert. Zum Schluß sei noch darauf hingewiesen, daß wir in der folgenden Prozedur zwei Standardfunktionen "Insert" und "Concat" zur Stringbearbeitung voraussetzen. Insert fügt einen Text in einen anderen Text ein und Concat reiht zwei Texte aneinander.

```

Procedure Verarbeitungsvorschrift_R1;
  Var Satzzeichenentscheidung_falsch,
      Weitere_Satzzeichenentscheidung_vorgesehen : Boolean;
  Function Letzte_Cursor_Position: Integer; {...}
  Function NeueNachricht: Text; {...}
  Function NeueNachricht_für_a: Text; {...}

```

```

Function NeueNachricht_für_b: Text; {...}
Function NeueNachricht_für_Auswahl: Text; {...}
Function NeueNachricht_für_falsch: Text; {...}
Function NeueNachricht_für_Auswertung: Text; {...}
Function Neue_Nachricht_für_Dialogende: Text;
Begin {Verarbeitungsvorschrift_R1}
Case Zustandsmarke.Phase of
Beschreibung
  Case Übergabemarke.Funktion of
    [Texteingabe]: Begin
      Zustandsmarke_S1.Benutzer.Name := Übergabemarke.Nachricht;
      Zustandsmarke_S1.Nachricht := NeueNachricht;
      Übergabemarke.Nachricht := Zustandsmarke_S1.Nachricht;
      Übergabemarke.Funktion := [Textausgabe]
    End;
    [Ausgabe_fortsetzen] :
      Begin
        Zustandsmarke_S1.Nachricht := NeueNachricht;
        Übergabemarke.Nachricht := Zustandsmarke_S1.Nachricht;
        Übergabemarke.Funktion := [Textausgabe]
      End;
    [Ja_Nein] : Begin
      If Übergabemarke.Nachricht = 'a'
        Then Zustandsmarke_S1.Nachricht := NeueNachricht_für_a
        Else Zustandsmarke_S1.Nachricht := NeueNachricht_für_b
      End
      Übergabemarke.Nachricht := Zustandsmarke_S1.Nachricht;
      Übergabemarke.Funktion := [Textausgabe, Tonausgabe]
    End;
    [Menüauswahl] :
      Begin
      If Übergabemarke.Nachricht = 'Ende'
        Then Begin
          Zustandsmarke_S1.Nachricht:=Neue_Nachricht_für_Dialogende;
          Übergabemarke.Nachricht := Zustandsmarke_S1.Nachricht;
          Übergabemarke.Funktion := [Textausgabe]
        End
        Else Begin
          Zustandsmarke_S1.Nachricht := NeueNachricht_für_Auswahl;
          Übergabemarke.Nachricht := Zustandsmarke_S1.Nachricht
          Übergabemarke.Funktion := [Textausgabe];
          Zustandsmarke_S1.Phase := Erklärung
        End
      End
    End; {case}
  Erklärung :
    Begin
      Zustandsmarke_S1.Nachricht := NeueNachricht;
      If Zustandsmarke_S1.Nachricht ≠ 'End_of_Erklärung'
        Then Begin
          Übergabemarke.Nachricht := Zustandsmarke_S1.Nachricht
          Übergabemarke.Funktion := [Textausgabe]
        End
      End
    End
  End

```

```

Else Begin
    Zustandsmarke_S1.Nachricht := NeueNachricht;
    Übergabemarke.Nachricht := Zustandsmarke_S1.Nachricht;
    Übergabemarke.Funktion := [Textausgabe, Inverse_Textausgabe, Cursor];
    Zustandsmarke_S1.Phase := Testsätze
End

End;
Testsätze :
Begin
    { Überprüfe Übergabemarke.Nachricht auf richtige Satzzeichenentscheidung :
    If Satzzeichenentscheidung_falsch
        Then Begin
            Zustandsmarke_S1.Benutzer.Fehler := Zustandsmarke_S1.Benutzer.Fehler + 1
            Übergabemarke.Funktion := [Tonausgabe];
            Zustandsmarke_S1.Nachricht := Insert ( NeueNachricht,
                Letzte_Cursor_Position, Zustandsmarke_S1.Nachricht )
            End
        Else Begin
            Zustandsmarke_S1.Benutzer.Richtig := Zustandsmarke_S1.Benutzer.Richtig + 1
            Übergabemarke.Funktion := [ ];
            Zustandsmarke_S1.Nachricht := Übergabemarke.Nachricht
            End;
    If Weitere_Satzzeichenentscheidung_vorgesehen
        Then Begin
            Übergabemarke.Nachricht := Zustandsmarke_S1.Nachricht;
            Übergabemarke.Funktion := Übergabemarke.Funktion +
                [Textausgabe, Inverse_Textausgabe, Cursor]
            End
        Else Begin
            If Satzzeichenentscheidung_falsch
                Then Übergabemarke.Nachricht :=
                    Concat (Zustandsmarke_S1.Nachricht, NeueNachricht_für_falsch)
                Else Übergabemarke.Nachricht := ' ';
            Zustandsmarke_S1.Nachricht := NeueNachricht;
            If Zustandsmarke_S1.Nachricht ≠ 'End_of_Testsätze'
                Then Begin
                    Übergabemarke.Nachricht := Concat ( Übergabemarke.Nachricht,
                        Zustandsmarke_S1.Nachricht);
                    Übergabemarke.Funktion := Übergabemarke.Funktion +
                        [Textausgabe, Inverse_Textausgabe, Cursor]
                    End
                Else Begin
                    Zustandsmarke_S1.Nachricht := NeueNachricht_für_Auswertung;
                    Übergabemarke.Nachricht := Zustandsmarke_S1.Nachricht;
                    Übergabemarke.Funktion := [Textausgabe];
                    Zustandsmarke_S1.Phase := Beschreibung
                    End
                End {else}
            End {Testsätze}
        End {case}
End; {Verarbeitungsvorschrift_R1}

```

```

Procedure Verarbeitungsvorschrift_R2;
  Begin
    Zustandsmarke_S1.Zustand := Beendet
  End;

```

```

Procedure Verarbeitungsvorschrift_M2;
  Begin
    Zustandsmarke_G1 := Dialog_beendet;
    Zustandsmarke_G2 := Dialog_ist_beendet
  End

```

Konfliktlösungsvorschriften

In unserem Modell (vgl. Abb. 11.12) stehen jeweils die Instanzen Ra1 und Ra2 und R1 und R2 in Konflikt miteinander, weil jeweils beide einen gemeinsamen Eingangskanal haben und entschieden werden muß, welche der beiden schalten soll oder ob beide gleichzeitig schalten müssen. Dazu ist eine Konfliktlösungsvorschrift nötig, aufgrund derer der Konflikt entschieden wird. Entsprechend unserer Vorüberlegungen formulieren wir - wieder in Pascal-ähnlicher Notation - eine Konfliktlösungsvorschrift für die Instanzen R1 und R2, die eine der beiden Instanzen auswählt und eine Konfliktlösungsvorschrift für Ra1 und Ra2, die auch paralleles Schalten der beiden Instanzen zuläßt.

```

Procedure Konfliktlösungsvorschrift_R1_R2;
  Begin
    If Übergabemarke.Funktion = [Dialogende]
      Then Verarbeitungsvorschrift_R2
      Else Verarbeitungsvorschrift_R1
  End;

```

```

Procedure Konfliktlösungsvorschrift_Ra1_Ra2;
  Begin
    If [Textausgabe,Tonausgabe] in Übergabemarke.Funktion
      Then Cobegin
        Verarbeitungsvorschrift_Ra1;
        Verarbeitungsvorschrift_Ra2
      Coend
    Else If [Textausgabe] in Übergabemarke.Funktion
      Then Verarbeitungsvorschrift_Ra1
      Else Verarbeitungsvorschrift_Ra2
  End

```

Da in Pascal eine parallele Durchführung von Anweisungen nicht möglich ist, haben wir in der Konfliktlösungsvorschrift_Ra1_Ra2 die Wortsymbole Cobegin und Coend eingeführt, die bedeuten sollen, daß die durch sie eingeschlossenen Anweisungen parallel durchzuführen sind und daß mit dem Programm erst dann fortgefahren werden kann, wenn beide abgeschlossen sind (Synchronisationsvorschrift).

12. Organisationsmodelle für den allgemeinen Dialograhmen

In Abschnitt 6.3. wurde der allgemeine Dialograhmen eingeführt, um für den Bereich CUU ein Instrument anzubieten, das einerseits Unabhängigkeit von Ein- und Ausgabegeräten und andererseits durch Integration von Anwendungsprogrammen auch Anwendungsunabhängigkeit gewährleisten soll. Zur Architektur des allgemeinen Dialograhmens wurde in Abschnitt 6.4. eine Hierarchie aus einem Netz von Arbeitsplatzrechnern, einem Schulrechner und einem Großrechner vorgeschlagen, die über die heutigen finanziellen und organisatorischen Möglichkeiten von Schulen zwar hinausgeht, prinzipiell aber den heute denkbaren Anwendungen, grundsätzlichen Forderungen und Wünschen und auch zukünftigen Entwicklungen in diesem Bereich - soweit heute erkennbar - gerecht wird.



Abb. 12.1.: Sternförmige Netztopologie

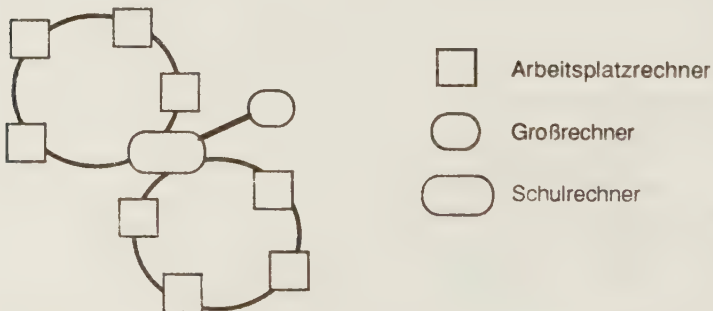


Abb. 12.2.: Ringförmige Netztopologie mit zentralem Vermittler

Die Überlegungen aus Abschnitt 6.3 sollen im folgenden konkretisiert werden indem ein Organisationsmodell in Form eines Petri-Netzes gebildet wird, das die

geforderten Eigenschaften des allgemeinen Dialograhmens auf der Konfigurationsebene nachbildet. Dabei ist die Organisationsebene als Modell des Dialogsystems, wie es logisch für die Benutzer, Anwender und Systementwickler konfiguriert ist, als unabhängig zu sehen von der technischen Realisierung der Konfiguration.

Es ist für unsere Sichtweise unerheblich, ob der allgemeine Dialograhmen sternförmig (Abb. 12.1.), ringförmig mit zentralem Vermittler (Abb. 12.2.) oder als Diffusionsnetz mit oder ohne Vermittler (Abb. 12.3. und Abb. 12.4.) realisiert wird. Wichtig ist allein die Tatsache, daß das zur Verfügung stehende Netz die für uns notwendige logische Hierarchie Arbeitsplatzrechner - Schulrechner - Großrechner zuläßt.

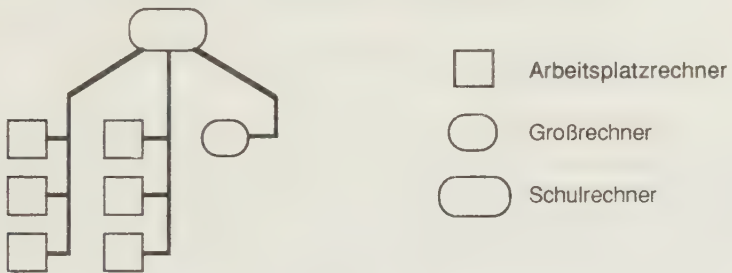


Abb. 12.3.: Diffusionsnetz mit zentralem Vermittler

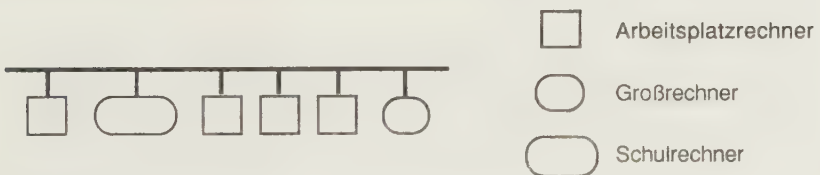


Abb. 12.4.: Diffusionsnetz ohne zentralen Vermittler

12.1. Ein Grundmodell für den allgemeinen Dialograhmen auf der Organisationsebene

Ein Grundmodell zur Darstellung der konzipierten Hierarchie - bestehend aus einem Arbeitsplatzrechner, einem Schulrechner und einem Großrechner - kann in Form eines Petri-Netzes durch Erweiterung des dialogfähigen Systems aus Abb. 11.3. gewonnen werden (vgl. Abb. 12.5.).

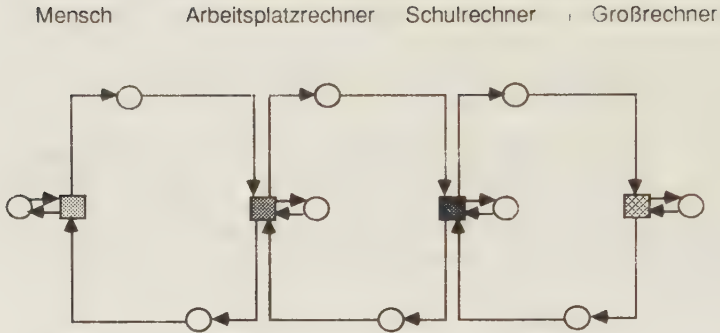


Abb. 12.5.: Grundmodell für den allgemeinen Dialograhmen auf der Organisationsebene

Dieses Modell zeigt zwar prinzipiell die hierarchischen Beziehungen zwischen den am Dialog beteiligten Instanzen, wird aber aus mehreren Gründen den Anforderungen zum allgemeinen Dialograhmen nicht gerecht. Der allgemeine Dialograhmen sieht unterschiedliche Ein- und Ausgabegeräte für einen Benutzer vor. Diese sind im Modell zu berücksichtigen. Erreicht wird dies durch Verfeinerung der zwischen Benutzer und Arbeitsplatzrechner vorgesehenen Kanäle in der in Abschnitt 11.3.1. beschriebenen Weise. Der allgemeine Dialograhmen soll einen Datenaustausch zwischen unterschiedlichen Arbeitsplatzrechnern ermöglichen. Ein entsprechendes Modell muß also Verbindungen zwischen Arbeitsplatzrechnern wiedergeben.

Die Kommunikation zwischen Benutzer und System hat im allgemeinen Dialograhmen eine zentrale Bedeutung. Dieser Aspekt muß bereits auf der Organisationsebene berücksichtigt werden, damit die Trennung zwischen Kommunikation und Verarbeitungsleistung (durch eine Anwendung) und damit die Anwendungsunabhängigkeit des allgemeinen Dialograhmens deutlich wird. Das Modell muß bereits auf dieser Ebene das bei den Systemmodellen notwendige Schalten von Instanzen berücksichtigen, d.h. die Komponenten des allgemeinen Dialograhmens müssen durch Instanzen in einer Weise nachgebildet werden, die beim Übergang zu markierten Petri-Netzen für Systemmodelle zu schaltfähigen Netzen führt.

12.2. Das Organisationsmodell für den allgemeinen Dialograhmen

Eine Erweiterung des Grundmodells zur Darstellung des allgemeinen Dialograhmens auf der Organisationsebene, das den genannten Kriterien genügt, ist in

geforderten Eigenschaften des allgemeinen Dialograhmens auf der Konfigurationsebene nachbildet. Dabei ist die Organisationsebene als Modell des Dialogsystems, wie es logisch für die Benutzer, Anwender und Systementwickler konfiguriert ist, als unabhängig zu sehen von der technischen Realisierung der Konfiguration.

Es ist für unsere Sichtweise unerheblich, ob der allgemeine Dialograhmen sternförmig (Abb. 12.1.), ringförmig mit zentralem Vermittler (Abb. 12.2.) oder als Diffusionsnetz mit oder ohne Vermittler (Abb. 12.3. und Abb. 12.4.) realisiert wird. Wichtig ist allein die Tatsache, daß das zur Verfügung stehende Netz die für uns notwendige logische Hierarchie Arbeitsplatzrechner - Schulrechner - Großrechner zuläßt.

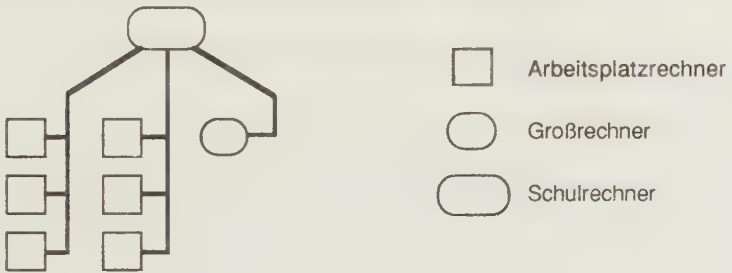


Abb. 12.3.: Diffusionsnetz mit zentralem Vermittler

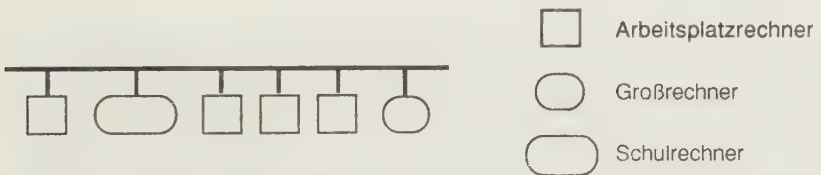


Abb. 12.4.: Diffusionsnetz ohne zentralen Vermittler

12.1. Ein Grundmodell für den allgemeinen Dialograhmen auf der Organisationsebene

Ein Grundmodell zur Darstellung der konzipierten Hierarchie - bestehend aus einem Arbeitsplatzrechner, einem Schulrechner und einem Großrechner - kann in Form eines Petri-Netzes durch Erweiterung des dialogfähigen Systems aus Abb. 11.3. gewonnen werden (vgl. Abb. 12.5.).

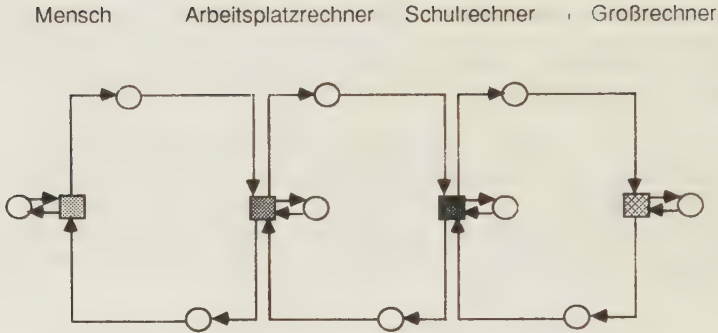


Abb. 12.5 : Grundmodell für den allgemeinen Dialograhmen auf der Organisationsebene

Dieses Modell zeigt zwar prinzipiell die hierarchischen Beziehungen zwischen den am Dialog beteiligten Instanzen, wird aber aus mehreren Gründen den Anforderungen zum allgemeinen Dialograhmen nicht gerecht. Der allgemeine Dialograhmen sieht unterschiedliche Ein- und Ausgabegeräte für einen Benutzer vor. Diese sind im Modell zu berücksichtigen. Erreicht wird dies durch Verfeinerung der zwischen Benutzer und Arbeitsplatzrechner vorgesehenen Kanäle in der in Abschnitt 11.3.1. beschriebenen Weise. Der allgemeine Dialograhmen soll einen Datenaustausch zwischen unterschiedlichen Arbeitsplatzrechnern ermöglichen. Ein entsprechendes Modell muß also Verbindungen zwischen Arbeitsplatzrechnern wiedergeben.

Die Kommunikation zwischen Benutzer und System hat im allgemeinen Dialograhmen eine zentrale Bedeutung. Dieser Aspekt muß bereits auf der Organisationsebene berücksichtigt werden, damit die Trennung zwischen Kommunikation und Verarbeitungsleistung (durch eine Anwendung) und damit die Anwendungsunabhängigkeit des allgemeinen Dialograhmens deutlich wird. Das Modell muß bereits auf dieser Ebene das bei den Systemmodellen notwendige Schalten von Instanzen berücksichtigen, d.h. die Komponenten des allgemeinen Dialograhmens müssen durch Instanzen in einer Weise nachgebildet werden, die beim Übergang zu markierten Petri-Netzen für Systemmodelle zu schaltfähigen Netzen führt.

12.2. Das Organisationsmodell für den allgemeinen Dialograhmen

Eine Erweiterung des Grundmodells zur Darstellung des allgemeinen Dialograhmens auf der Organisationsebene, das den genannten Kriterien genügt, ist in

Abb. 12.6. dargestellt. Es bildet die beiden mittleren Komponenten der Hierarchie jeweils durch vier Instanzen, die Großrechner-Komponente durch zwei Instanzen nach.

Alle drei die Rechner betreffenden Ebenen der Hierarchie verfügen jetzt über eine Kommunikations- und eine Verarbeitungsinstanz. Während die Verarbeitungsinstanz mit ihrem Zustandskanal die Rechnerleistung in Bezug auf das eigentliche Anwendungsprogramm darstellt, modelliert die Kommunikationsinstanz diejenigen Verarbeitungsschritte, die für den Mensch-Rechner-Dialog notwendig sind. Hierzu zählen z.B. das Aufbereiten von Ein- und Ausgaben und das Analysieren von Eingaben (z.B. welche Komponente der Hierarchie durch eine Eingabe angesprochen wird). Die "Kommunikationsinstanz" unseres Modells entspricht der "MCS" von Balzert /Balzert86/.

Die beiden mittleren Komponenten verfügen darüberhinaus noch über zwei Instanzen, die die Verbindung zum jeweiligen Nachbarn innerhalb der Hierarchie schaffen. Durch sie werden Daten zur nächsten Komponente übergeben, bzw. von der nachgelagerten Komponente entgegengenommen.

Ein Datenaustausch zwischen einzelnen Arbeitsplatzrechnern wird schließlich möglich durch Aufnahme einer Datenübertragungsinstanz im Anschluß an die Kommunikationsinstanz des Arbeitsplatzrechners. Die im Modell strichliert dargestellten Pfeile verbinden jeweils zwei Petri-Netze der in Abb. 12.6. dargestellten Art und stellen keine Relationen innerhalb eines Netzes dar. Erst bei Definition eines übergeordneten Petri-Netzes, das entsprechend der Anzahl der Arbeitsplatzrechner mehrere solcher logischer Hierarchien zusammenfaßt, können diese Verbindungen als Elemente der Relation definiert werden.

Bei dem Entwurf des Modells gemäß Abb. 12.6. sind wir von Kommunikationsinstanzen für jede Komponente der Hierarchie ausgegangen. Nimmt man die Forderungen zur Integration von Anwendungsprogrammen zur Schaffung einer einheitlichen Benutzerschnittstelle innerhalb des allgemeinen Dialograhmens, so ist anstelle mehrerer Kommunikationsinstanzen die Definition einer zentralen solchen Instanz naheliegend, mit der alle Komponenten der Hierarchie verbunden sind (vgl. Abb. 12.7.).

Allerdings ist dann die eigentliche Hierarchie des allgemeinen Dialograhmens nicht mehr im Modell wiedergegeben. Der Nachrichtenfluß führt jetzt von der zentralen Kommunikationsinstanz zu den einzelnen Komponenten des Dialograhmens. Dadurch erhalten wir die Kommunikationsinstanz als zentrale Koordinationseinheit des Systems. Gerade solch eine zentrale Koordination ist im allgemeinen Dialograhmen jedoch nicht vorgesehen, sondern z.B. der Schulrechner ist dem Arbeitsplatzrechner bewußt nachgelagert und Dialogpartner für den Benutzer im Falle von Aufgaben, die nicht (mehr) vom Arbeitsplatzrechner erledigt werden können, bzw. sollen. Diese Konzeption der

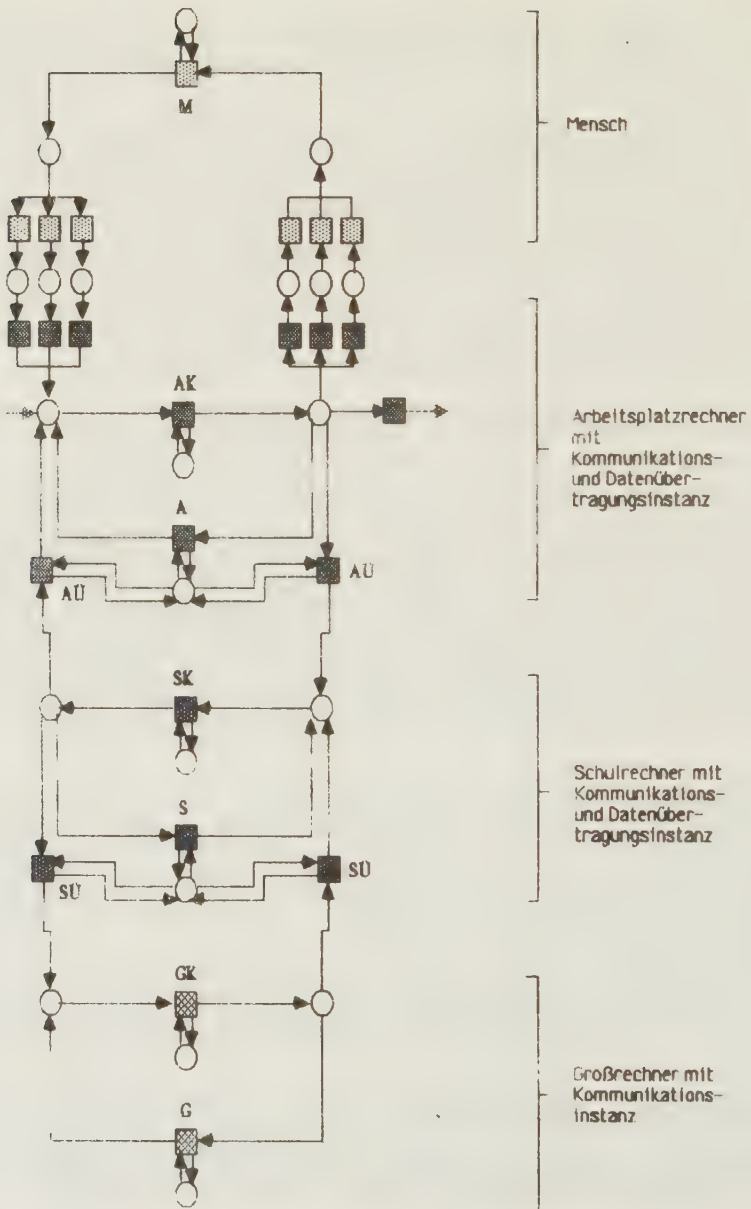


Abb. 12.6.: Modell des allgemeinen Dialograhmens auf der Organisationsebene

Die Bedeutung der Abkürzungen in Abb. 12.6 ergibt sich wie folgt:

- M : Mensch
- AK : Arbeitsplatzrechner-Kommunikationsinstanz
- AU : Arbeitsplatzrechner-Übertragungsinstanz
- A : Arbeitsplatzrechner-Verarbeitungsinstanz
- SK : Schulrechner-Kommunikationsinstanz
- SÜ : Schulrechner-Übertragungsinstanz
- S : Schulrechner-Verarbeitungsinstanz
- GK : Großrechner-Kommunikationsinstanz
- G : Großrechner-Verarbeitungsinstanz

hierarchischen Beziehung zwischen den Komponenten gibt das Modell aus Abb. 12.7. nicht wieder.

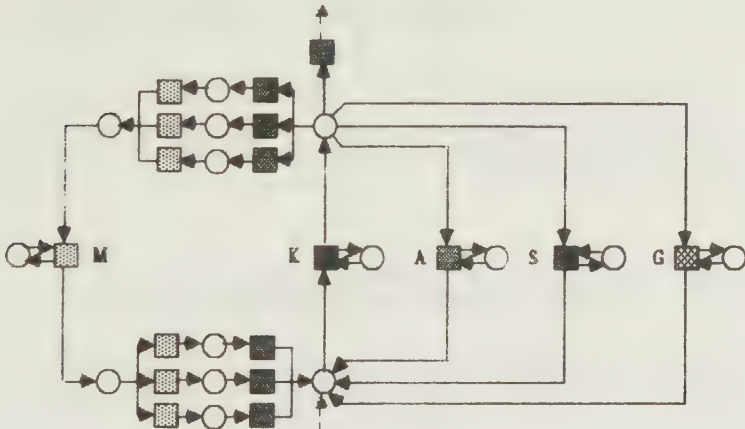


Abb. 12.7 Organisationsmodell des allgemeinen Dialograhmens mit zentraler Kommunikationsinstanz zur Systemkoordination

Die Möglichkeit, nur zwischen Benutzer und Arbeitsplatzrechner eine Kommunikationseinheit vorzusehen und ansonsten die Hierarchie aus Abb. 12.6. beizubehalten - dargestellt in Abb. 12.8 - gibt die notwendigen Systemeigenschaften ebenfalls nicht korrekt wieder. Selbst bei zentraler Kontrolle der Kommunikation werden auf der Schulrechner- und Großrechner-Ebene Verarbeitungsschritte zur Analyse und Aufbereitung von den Dialog betreffenden Aktionen nötig. Als Beispiel wäre die Feststellung der -

bezogen auf eine Anwendung - eingegebenen Funktion in Anwendungsdialogen zu nennen oder auch die Analyse, ob die einlaufenden Nachrichten weiterzuleiten oder am Ort zu verarbeiten sind.

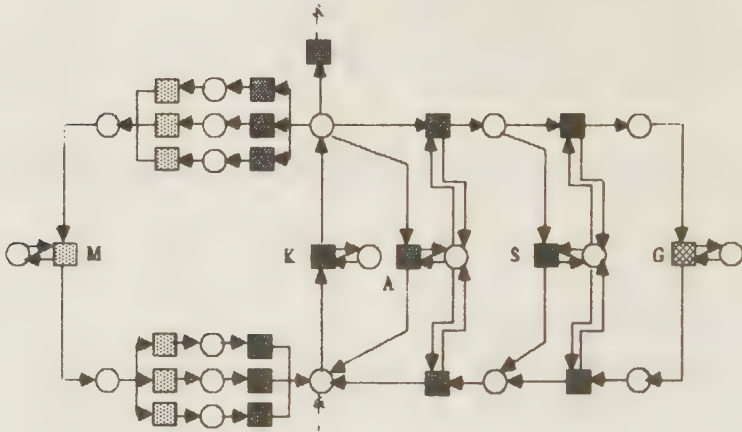


Abb. 12.8.: Organisationsmodell des allgemeinen Dialograhmens mit zentraler Kommunikationsinstanz ohne Systemkoordination

Die Abkürzungen in den Abbildungen 12.7 und 12.8 bedeuten:

- M : Mensch
- K Kommunikationsinstanz
- A Arbeitsplatzrechner-Instanz
- S Schulrechner-Instanz
- G Großrechner-Instanz

Wir werden deshalb für das Folgende das Modell aus Abb. 12.6. voraussetzen und für jede Rechnerkomponente eine Kommunikationsinstanz vorsehen. Dabei wird hinsichtlich der Benutzerschnittstelle der dem Benutzer am nächsten gelegenen Kommunikationsinstanz des Arbeitsplatzrechners besondere Bedeutung zukommen.

13. Systemmodelle für den allgemeinen Dialograhmen

Bei der Definition von Systemmodellen für den allgemeinen Dialograhmen wollen wir zunächst vom Organisationsmodell des letzten Kapitels ausgehen und dieses kombinieren mit den auf Systemebene grundlegenden Klassen Dialogeröffnung, normaler Dialog und Dialogbeendigung. Wir erhalten dadurch das "erste Systemmodell des allgemeinen Dialograhmens" aus Abschnitt 13.1. Dieses Modell wird wegen einiger Unzulänglichkeiten dann in den folgenden Abschnitten verfeinert.

Das "zweite Systemmodell" trennt bei der Kommunikation Eingaben und Ausgaben und unterscheidet weiter zwischen Kommunikation mit dem Benutzer und innerhalb der Rechner. Das Modell berücksichtigt darüberhinaus Fenster als zentrale Elemente der Benutzerschnittstelle und sieht außerdem Instanzen zur Gewährleistung von Geräteunabhängigkeit des Dialograhmens vor.

Waren die Veränderungen für dieses zweite Systemmodell einfache Verfeinerungen des ersten Modells, so bewirkt die Berücksichtigung der abstrakten Dialogform Unterbrechung im "dritten Systemmodell des allgemeinen Dialograhmens" eine völlige Reorganisation der Kanal-Instanzen-Beziehungen, damit die möglichen Dialogschritte im darstellenden Netz wiedergegeben werden können.

13.1. Ein erstes Systemmodell mit den Klassen Dialogeröffnung, Dialogbeendigung und normaler Dialog

Auf der Basis des im letzten Kapitel definierten Organisationsmodells (vgl. Abb. 12.6) schaffen wir ein erstes Systemmodell für den allgemeinen Dialograhmen, indem wir die für diese Abstraktionsebene grundlegenden Klassen Dialogeröffnung, normaler Dialog und Dialogbeendigung durch Kanal-Instanzen-Kombinationen im Modell unterscheiden.

Dabei wollen wir dem Modell die Idee zugrundelegen, einem Benutzer nur diejenigen Rechner als Kommunikationspartner zur Verfügung zu stellen, die er in seinem aktuellen Dialog tatsächlich anspricht: Zunächst findet eine Dialogeröffnung nur mit dem Arbeitsplatzrechner statt. Dieser ist dem Benutzer am nächsten und wird für den Dialog immer benötigt. Betrifft eine Benutzerfunktion im normalen Dialog nicht den Arbeitsplatzrechner sondern eine nachgelagerte Systemkomponente, so muß der Dialog mit dieser erst zu diesem Zeitpunkt eröffnet werden. Der Arbeitsplatzrechner dient dann zur Datendurchreiche und ebenso, falls die Benutzerfunktion den Großrechner betrifft, die Schulrechner-Komponente. Der Zustand der Eröffnung bleibt solange erhalten,

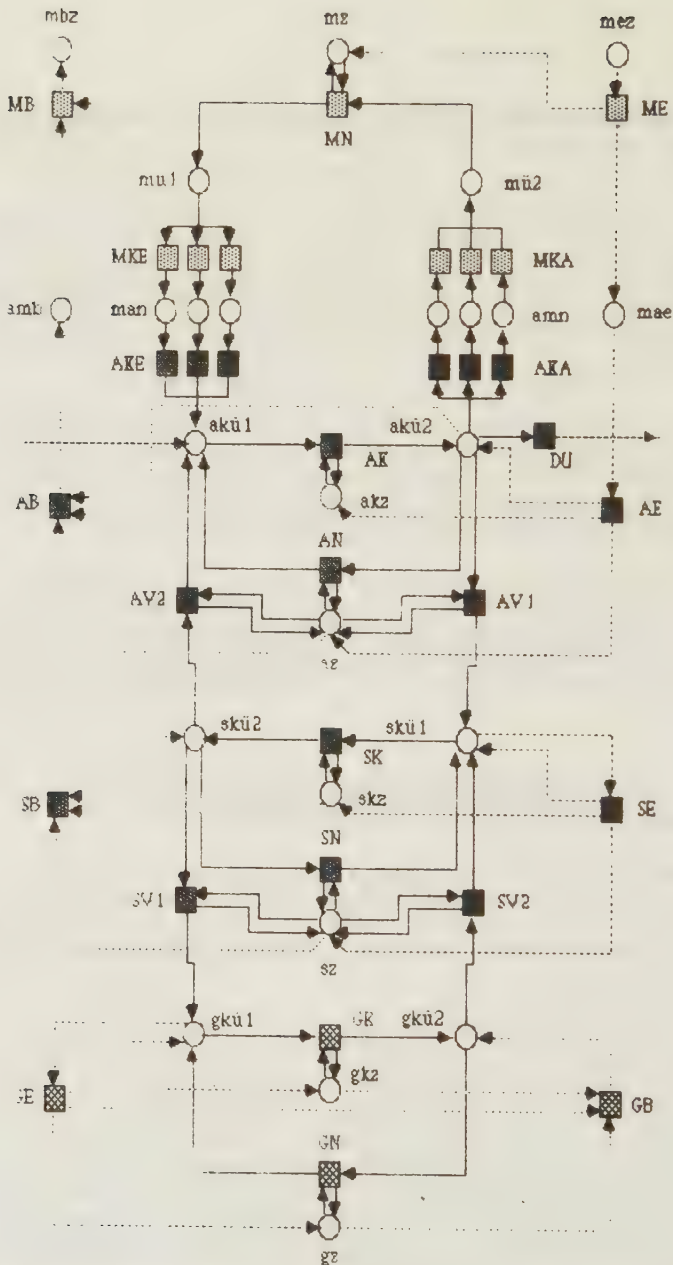


Abb. 13.1: Das "Erste Systemmodell" für den allgemeinen Dialograhmen

wie tatsächlich die Dialogschritte sich auf diesen Rechner beziehen. Das Ansprechen einer anderen Systemkomponente durch den Benutzer führt - sofern der betrachtete Rechner nicht zur Datendurchreiche benötigt wird - zur Beendigung des Dialogs mit diesem Rechner

Dadurch paßt sich das System flexibel den jeweiligen Anforderungen an. Die in der Regel stattfindende Kommunikation zwischen Benutzer und Arbeitsplatzrechner wird durch ständige Verfügbarkeit des Rechners ohne Einschränkungen möglich. Ein höherer technischer und Software-Aufwand bei Zugriffen auf den Schulrechner, sowie zusätzlich möglicherweise erhöhte Kosten (z. B. durch eine Wählleitung) bei Benutzung des Großrechners fallen nur dann an, wenn der Dialog diese Systemkomponente betrifft.

Diese Flexibilität auf Seiten des Systems darf nun nicht einhergehen mit erhöhtem Aufwand für den Benutzer. Zwar wird sich beim erstmaligen Ansprechen einer nachgelagerten Systemkomponente unter Umständen durch die notwendige Eröffnung die Antwortzeit erhöhen, mit der eigentlichen Eröffnung hat der Benutzer jedoch nichts zu tun. Sie wird implizit bei Feststellung, daß ein bestimmter Rechner als Dialogpartner benötigt wird durchgeführt.

Abbildung 13.1 gibt das Petri-Netz für das erste Systemmodell des allgemeinen Dialograhmens wieder, wobei Kanäle mit Klein- und Instanzen mit Großbuchstaben wie folgt bezeichnet wurden:

mez	:	m enschlicher E röffnungszustandskanal
mz	:	m enschlicher Z ustandskanal
mbz	:	m enschlicher B eendigungszustandskanal
ME	:	m enschliche E röffnungsinstanz
MN	:	m enschliche Instanz für n ormalen Dialog
MB	:	m enschliche B eendigungsinstanz
mae	:	M ensch- A rbeitsplatzrechner-Kanal zur E röffnung
amb	:	A rbeitsplatzrechner- M ensch-Kanal zur B eendigung
mü1, mü2	:	m enschliche Ü bertragungskanäle
MK1, MK2	:	m enschliche K ommunikationsinstanzen
man	:	M ensch- A rbeitsplatzrechner-Kanal für n ormalen Dialog
amn	:	A rbeitsplatzrechner- M ensch-Kanal für n ormalen Dialog
AKE	:	A rbeitsplatzrechner- K ommunikationsinstanz für E ingaben
AKA	:	A rbeitsplatzrechner- K ommunikationsinstanz für A usgaben
(Die letzten sechs Bezeichnungen stehen stellvertretend für eine theoretisch beliebig große Zahl von E/A - Geräten. In Abb. 13.1. wurden drei E/A - Geräte dargestellt)		
akü1, akü2	:	A rbeitsplatzrechner - K ommunikations - Ü bertragungskanäle
AK	:	A rbeitsplatzrechner - K ommunikationsabwicklungsinstanz
akz	:	A rbeitsplatzrechner - K ommunikationsabwicklungs - Z ustandskanal

AE	: Arbeitsplatzrechner - Eröffnungsinstanz
AB	: Arbeitsplatzrechner - Beendigungsinstanz
AN	: Arbeitsplatzrechner - Instanz für normalen Dialog
AV1, AV2	: Arbeitsplatzrechner - Verbindungsinstanz zu nachgelagerten Systemkomponenten
az	: Arbeitsplatzrechner - Zustandskanal

Die Bezeichnungen der Instanzen und Kanäle für den Schul- und den Großrechner folgen mit "S" für Schulrechner und "G" für Großrechner der obigen Systematik. Relationen im Zusammenhang mit der Eröffnung wurden gestrichelt mit der Dialogbeendigung gepunktet dargestellt. Die mit DÜ bezeichnete Instanz dient der Datenübertragung zwischen verschiedenen Arbeitsplatzrechnern (vgl. Abschnitt 12.2).

Wir wollen die geschilderten Eigenschaften anhand der Abbildung 13.1 konkretisieren und nachvollziehen und dabei die Funktion einzelner Instanzen und Kanäle beschreiben, allerdings ohne hier bereits auf Algorithmen für Instanzen und Markierungen von Kanälen im Einzelnen einzugehen (siehe hierzu das nächste Kapitel).

Eine zentrale Bedeutung im Modell aus Abb. 13.1. besitzen die Kommunikationsinstanzen AK, SK und GK. Jede Nachricht vom Benutzer für das System, vom System für den Benutzer und innerhalb des Systems wird von ihnen bearbeitet, analysiert und "auf den richtigen Weg gebracht". Ausgangspunkt ist dabei eine Markierung und eine Nachricht bei akü1. skü1 oder gkü1. aufgrund derer die jeweilige Kommunikationsinstanz die zuständige Nachfolgeinstanz ermittelt. Z. B. im Falle von AK kann

- eine Nachricht vom Benutzer kommen und für den Arbeitsplatzrechner bestimmt sein; dann ist die Verarbeitungsinstanz AN des Arbeitsplatzrechners Nachfolgeinstanz;
- eine Nachricht vom Benutzer kommen und für eine nachgelagerte Systemkomponente bestimmt sein; dann ist die Verbindungsinstanz AV1 des Arbeitsplatzrechners zum Schulrechner Nachfolgeinstanz;
- eine Nachricht vom Benutzer kommen und zur Datenübertragung an einen anderen Benutzer bestimmt sein; dann ist die Übertragungsinstanz DÜ des Arbeitsplatzrechners Nachfolgeinstanz;
- eine Nachricht von einem anderen Benutzer kommen und für den Benutzer bestimmt sein; dann ist eine Ausgabeinstanz AKA zuständig und Nachfolgeinstanz;
- eine Nachricht vom Arbeitsplatzrechner (AN) kommen und für den Benutzer bestimmt sein; die Nachfolgeinstanz ist dann wieder eine Ausgabeinstanz AKA
- eine Nachricht über die Verbindungsinstanz AV2 des Arbeitsplatzrechners vom Schulrechner kommen und für den Benutzer bestimmt sein; dann ist wiederum AKA Nachfolgeinstanz;
- eine Nachricht vom Arbeitsplatzrechner (AN) kommen und für eine nachge-

lagerte Komponente bestimmt sein, jetzt ist die Verbindungsinstanz AV1 als Nachfolgeinstanz zuständig;

- eine Nachricht über die Verbindungsinstanz AV2 vom Schulrechner kommen und für den Arbeitsplatzrechner, also AN bestimmt sein; AN ist dann Nachfolgeinstanz;

Mit diesen Festlegungen wollen wir einige unterschiedliche Dialogschritte am Modell nachvollziehen.

Die eigentliche **Dialogeröffnung** durch den Benutzer bewirkt eine Nachricht und Markierung auf dem Kanal mae. Daraufhin wird AE aktiv und versorgt die Zustandskanäle akz und az mit Marken (Laden von Dateien, Starten von Programmen) und macht dadurch den Arbeitsplatzrechner betriebsbereit. Weiter sendet AE eine Nachricht (und Marke) an akü2, die über eine Ausgabeinstanz AKA an den Benutzer weitergeleitet wird und die Meldung über die Betriebsbereitschaft beinhaltet (Saathoff82/; Dialogeröffnungssquittung).

Innerhalb des **normalen Dialogs** wird eine vom Arbeitsplatzrechner zu bearbeitende und vom Benutzer ausgelöste Funktion zunächst über die Eingabefolge mü1 - MK1 - man - AKE nach akü1 gesendet und dann von AK bearbeitet. AK bestimmt durch entsprechende Markierung auf akü2 die Instanz AN als Nachfolgeinstanz, woraufhin die Verarbeitungsinstanz AN des Arbeitsplatzrechners die Funktion bearbeiten kann. Eine Antwort von AN an den Benutzer wird dann wieder bei akü1 abgelegt. Die Kommunikationsinstanz AK wird aktiv, bestimmt eine Ausgabeinstanz AKA als Nachfolgeinstanz und leitet die Nachricht an akü2 weiter. Über AKA, amn, MK2 und mü2 gelangt diese dann zur menschlichen Verarbeitungsinstanz MN

Ist während des normalen Dialogs **eine vom Benutzer ausgelöste Funktion durch den Schulrechner zu bearbeiten**, so gelangt die Nachricht zunächst wieder nach akü1. AK bestimmt nach der Feststellung, daß nicht der Arbeitsplatzrechner als Dialogpartner zuständig ist, AV1 als Nachfolgeinstanz und setzt auf seinem Zustandskanal eine Markierung, daß der Dialog mit der Schulrechner - Komponente eröffnet wird. Die von AK auf akü2 weitergeleitete Nachricht wird von AV1 aufgenommen und bearbeitet. Dabei stellt AV1 mithilfe von az fest, daß der Dialog mit dem Schulrechner zu eröffnen ist und setzt durch eine Markierung auf skü1 die Instanz SE als Nachfolgeinstanz fest. Die vom Benutzer ausgegangene Nachricht wird nach skü1 weitergeleitet. Die Eröffnungsinstanz SE markiert daraufhin die Zustandsmarken skz und sz und macht dadurch den Schulrechner betriebsbereit. Die Betriebsbereitschaft wird auf skü1 durch Festlegung von SK als Nachfolgeinstanz angezeigt. Mit dieser Markierung wird auch die von AV1 übergebene Nachricht bei skü1 deponiert. Diese kann jetzt von SK aufgenommen und bearbeitet werden, wobei die Nachricht mit einer Markierung, daß SN - die Verarbeitungsinstanz des Schulrechners - Nachfolgeinstanz ist, an skü2 übergeben wird. Eine Antwort vom Schulrechner an den Benutzer als Folge der Arbeit von SN gelangt dann wieder

nach skü1, von wo sie durch SK aufgenommen und bearbeitet wird. Als Ergebnis wird AV2 als Nachfolgeinstanz bestimmt und eine entsprechende Markierung mit der Nachricht nach skü2 übergeben. Über AV2 - akü1 - AK - akü2 - AKA - amn - mk2 - mü2 gelangt die Nachricht dann zur Verarbeitungsinstanz des Menschen.

Dessen Reaktion führt über mü1 - MK1 - man - AKE zu einer Nachricht bei akü1 die wieder von AK bearbeitet wird. Für den Fall, daß AK feststellt, daß die Benutzerfunktion wiederum nicht für den Arbeitsplatzrechner bestimmt ist, wird nur AV1 als Nachfolgeinstanz eingestellt und die Nachricht nach akü2 weitergegeben. Da der Dialog mit der Schulrechner - Komponente eröffnet ist - für AV1 - aus der Markierung von az ersichtlich - bestimmt AV1 die Kommunikationsinstanz SK des Schulrechners als Nachfolgeinstanz und übergibt die Nachricht an skü1. Hier wird sie wieder in gewohnter Weise bearbeitet (s. o). Wenn AK jedoch feststellt, daß die Benutzerfunktion für den Arbeitsplatzrechner bestimmt ist, so wird vor Bearbeitung der Funktion durch AN der Dialog mit dem Schulrechner beendet.

Dies erfolgt in der Form, daß die Instanz AK die eigentliche Benutzernachricht bei akz zwischenspeichert, dort ebenfalls vermerkt, daß der Dialog mit der Schulrechner-Komponente beendet wird und AV1 als Nachfolgeinstanz bestimmt, mit der Nachricht, die Dialogbeendigung für den Schulrechner einzuleiten. AV1 legt daraufhin SK als Nachfolgeinstanz fest und übergibt eine entsprechende Markierung an skü1.

SK überprüft anhand seines Zustandskanals, ob ein Dialog mit dem Großrechner angemeldet ist. Ist dies der Fall, so muß vor Abschluß des Dialogs mit dem Schulrechner zunächst der Dialog mit dem Großrechner beendet werden. Dazu wird SV1 mit entsprechender Markierung und Attributierung als Nachfolgeinstanz bestimmt. Ist durch GB der Dialog mit dem Großrechner beendet worden, so gelangt eine entsprechende Nachricht über SV2 wieder zu der Instanz SK, die daraufhin SB zur Beendigung des Dialogs als Nachfolger bestimmt.

Ist kein Dialog mit dem Großrechner eröffnet, so wird von SK die Instanz SB ohne die genannten Zwischenschritte angesprochen. Jetzt zieht die Beendigungsinstanz SB die Marken von skz und sz ab und übergibt mit der Festlegung von AV2 als Nachfolgeinstanz die Nachricht, daß der Schulrechner nicht mehr betriebsbereit ist, an skü2. Die Instanz AV2 nimmt die Nachricht auf, vermerkt bei az, daß der Dialog mit dem Schulrechner beendet ist und übergibt die Nachricht an den Kanal akü1. Die Instanz AK nimmt daraufhin die zwischengespeicherte Nachricht aus dem Zustandskanal akz, übergibt sie mit der Festlegung von AN als Nachfolgeinstanz an akü2 und vermerkt auf akz, daß nur der Arbeitsplatzrechner betriebsbereit ist.

Eine **Dialogbeendigung durch den Benutzer** führt wieder zu einer Nachricht auf dem Kanal akü1 von wo sie durch AK verarbeitet wird. Ist zu

diesem Zeitpunkt der Dialog nur mit dem Arbeitsplatzrechner eröffnet, wird die Beendigungsinstanz AB als Nachfolgeinstanz gesetzt, die die Marken von az und akz abzieht und bei dem Kanal amb das Dialogende vermerkt. Wenn allerdings noch mit einer nachgelagerten Komponente ein Dialog besteht, so muß in der oben geschilderten Weise zuerst dort der Dialog beendet werden, bevor die Instanz AB arbeiten darf.

Eine Dialogeröffnung und Beendigung mit der Großrechner-Komponente erfolgt mithilfe des Schulrechners in der selben Weise, wie oben für den Schulrechner beschrieben. Ebenso folgt der normale Dialog mit dem Großrechner den oben beschriebenen Regeln, wie auch Dialogschritte zwischen den einzelnen Rechnern, die möglicherweise zur Erledigung von Benutzeranforderungen notwendig werden können.

Nachrichten von anderen Benutzern treffen innerhalb des Netzes bei akü1 ein. Für ihre Bearbeitung durch das Netz sind unterschiedliche Vorgehensweisen denkbar. So kann die Nachricht sofort durch die Instanz AK an den Benutzer weitergeleitet werden, wenn sie "im Netz eintrifft". Dies kann parallel zur Arbeit des Systems geschehen oder auch zu einer Unterbrechung der Schaltvorgänge im Netz führen. Ebenso ist eine Zwischenspeicherung der Nachricht bei akz und spätere Ausgabe an den Benutzer zusammen mit einer Systemnachricht möglich. Wir wollen uns innerhalb unserer weiteren Betrachtungen nur auf ein Netz - also eine Systemhierarchie - konzentrieren und vernachlässigen deshalb im folgenden nähere Untersuchungen der notwendigen Vorgänge im Falle eines Nachrichtenaustausches zwischen Benutzern

13.2. Ein zweites Systemmodell für den allgemeinen Dialograhmen zur Berücksichtigung von Geräte- und Anwendungsunabhängigkeit

Das Modell aus Abb. 13.1. sieht für die Benutzerschnittstelle die Instanz AK und mittelbar auch die Instanzen SK und GK vor. Berücksichtigen wir die in Teil II der Arbeit genannten Anforderungen hinsichtlich der Benutzerfreundlichkeit des allgemeinen Dialograhmens, wie Unterscheidung von Dialogformen und -typen, unterschiedliche Konkretisierungen und Realisierungen von Dialogformen und Anwendungs- und Geräteunabhängigkeit, so reicht für eine übersichtliche Nachbildung der Benutzerschnittstelle das bisherige Modell nicht aus. Wir werden deshalb im folgenden einige Verfeinerungen des Modells beschreiben, wobei wir uns in diesem Abschnitt auf Darstellung der Aspekte der Geräteunabhängigkeit, der Berücksichtigung von Fenstern als zentralem Element der Benutzerschnittstelle und der Möglichkeit unterschiedlicher Konkretisierungen und Realisierungen von Dialogformen beschränken. Die übrigen Aspekte werden dann im nächsten Abschnitt wieder aufgegriffen.

Bezüglich Geräteunabhängigkeit von Software-Systemen liegt seit einigen Jahren für die grafische Datenverarbeitung der DIN-Entwurf GKS vor. Dieses System stellt anwendungs- und geräteunabhängig eine Auswahl von Grundfunktionen zur Verfügung, durch die grafische Darstellungen manipuliert und verwaltet werden können. Die Basis dieses Systems, die abstrakten E/A-Funktionen, haben wir in Kapitel 10 beschrieben. Außer der Bearbeitung von rein grafischen Elementen beziehen sich die abstrakten Funktionen auch auf textuelle Information und bieten insbesondere durch eine Reihe von Attributfunktionen für die Textbearbeitung und -verarbeitung komfortable Möglichkeiten. Darüberhinaus ist durch GKS u. a. auch die Handhabung von Fenstern und Darstellungsfeldern genormt.

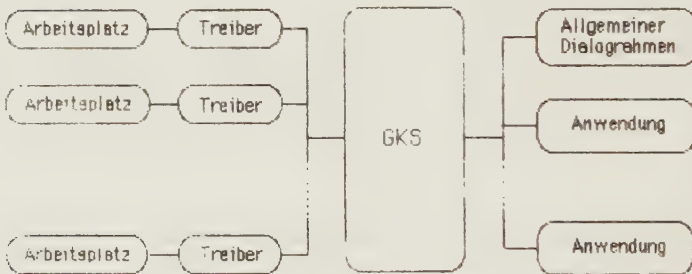


Abb. 13.2. Verbindung von GKS und dem allgemeinen Dialograhmen aus Sicht von GKS

Insgesamt stellt sich GKS nicht als ein reines Grafik-System dar, sondern muß als universelles geräteunabhängiges und interaktives System betrachtet werden, das beliebige Anwendungen unterstützt. Wir integrieren deshalb GKS in den allgemeinen Dialograhmen und sorgen dadurch für die gewünschte Geräteunabhängigkeit. Im Sinne von GKS ist der allgemeine Dialograhmen eine mögliche Anwendung des ansonsten anwendungsunabhängigen Systems (vgl. Abb. 13.2.). Im Sinne des allgemeinen Dialograhmens ist GKS ein für die Geräteunabhängigkeit zuständiger Bestandteil des ansonsten übergeordneten Systems (vgl. Abb. 13.3).

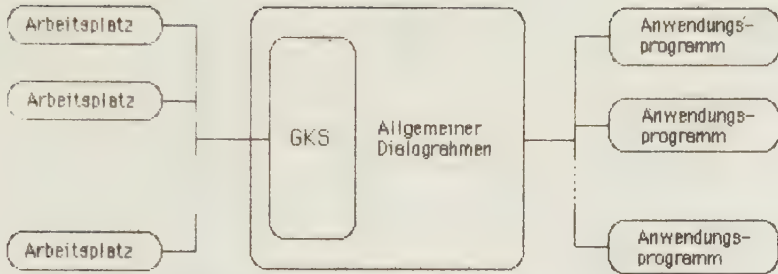


Abb. 13.3. GKS als Bestandteil des allgemeinen Dialograhmens

Innerhalb des zweiten Systemmodells für den allgemeinen Dialograhmen (vgl. Abb. 13.4.) berücksichtigen wir GKS durch zwei Instanzen mit gemeinsamem Zustandskanal, wobei eine Instanz die GKS-Funktionen für die Eingabe, die zweite die für die Ausgabe nachbildet.

(AKGE : Arbeitsplatzrechner **K**ommunikationsabwicklung-GKS-**E**ingabe

AKGA : Arbeitsplatzrechner **K**ommunikationsabwicklung-GKS-**A**usgabe

akzg : Arbeitsplatzrechner-Kommunikationsabwicklung-Zustandskanal für GKS)

Diese Trennung bei GKS ist in Zusammenhang zu sehen mit der generellen Trennung von Eingabeaktivitäten und Ausgabeaktivitäten bei der Kommunikationsabwicklung innerhalb dieses neuen Modells für den allgemeinen Dialograhmen. Eingabeaktivitäten sind dabei Funktionen bzw. Nachrichten, die vom Benutzer in das System bzw. von einer dem Benutzer näheren Komponente zu einer Nachbarkomponente eingegeben werden. Ausgabeaktivitäten bezeichnen den entgegengesetzten Nachrichtenfluß. Mit der Unterscheidung entzerren wir im Modell die vielfältigen Aufgaben der Kommunikationsinstanzen AK, SK und GK aus Abb. 13.1. und können so unterschiedliche Konkretisierungen und Realisierungen von Dialogformen besser berücksichtigen.

(AK1, AK2, AK3, AK4

: Arbeitsplatzrechner-Kommunikationsabwicklungsinstanzen

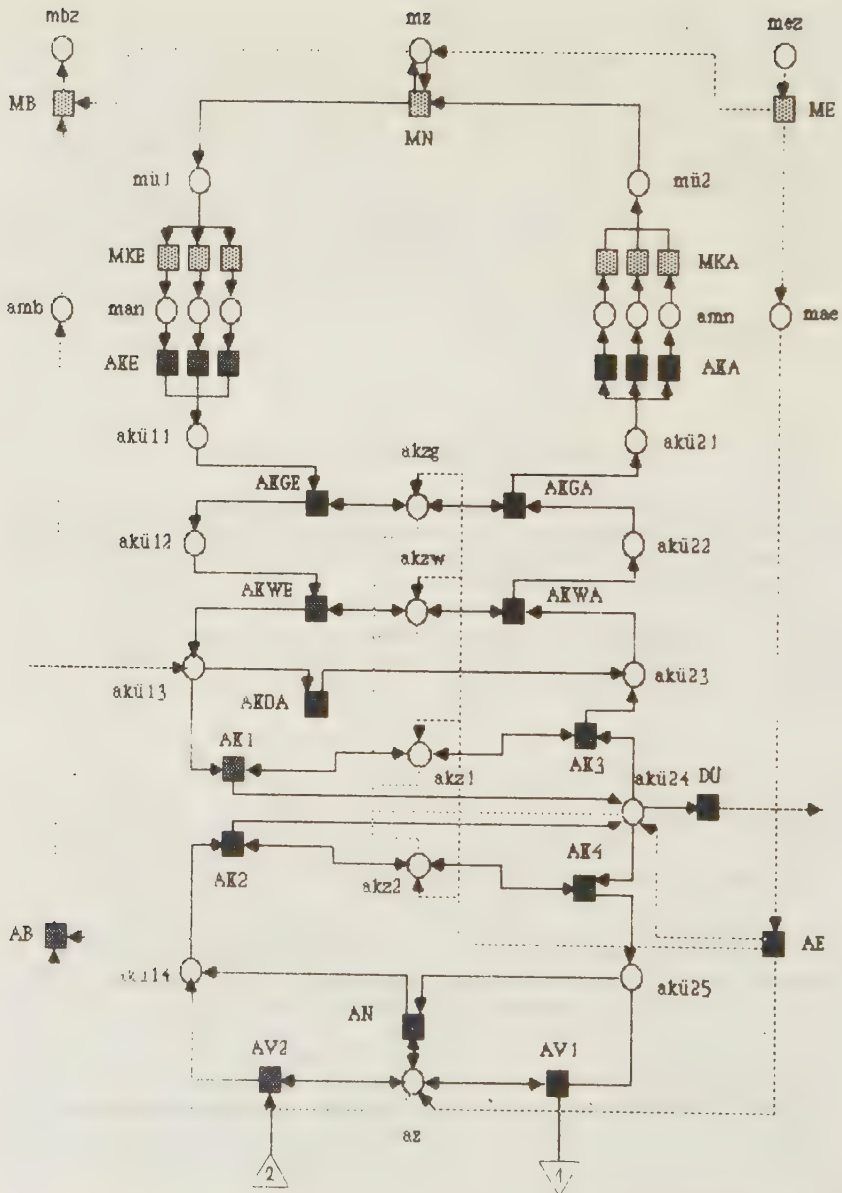
akz1, akz2, akz3, akz4

: Arbeitsplatzrechner-Kommunikationsabwicklungszustandskanäle

akü11, akü12, akü13, akü14

Arbeitsplatzrechner-Kommunikationsab-

akü21, akü22, akü23, akü24, akü25 : wicklungs-Übergabekanäle



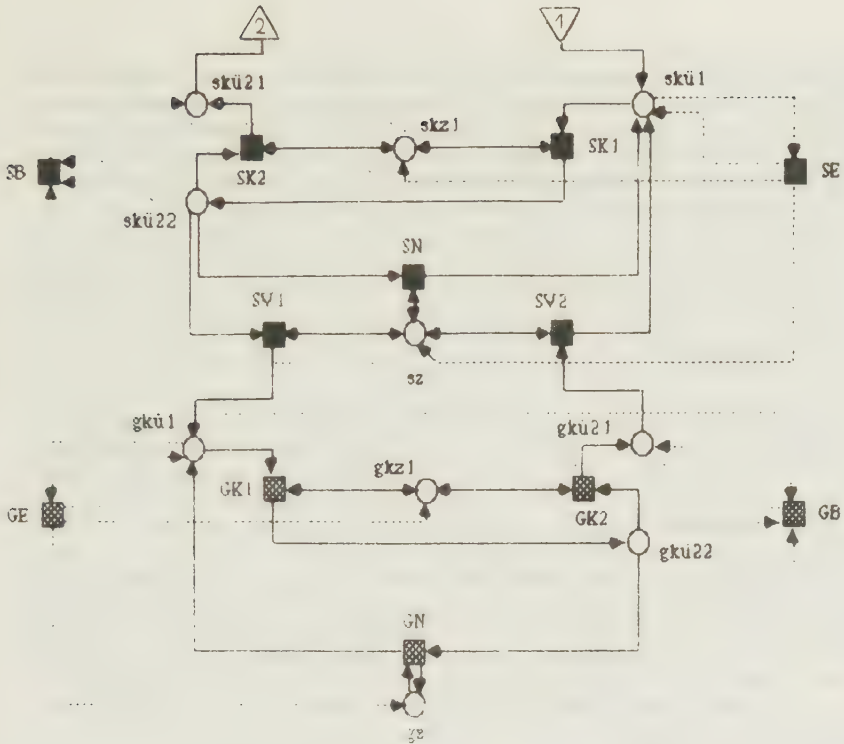


Abb. 13.4: Das "Zweite Systemmodell" für den allgemeinen Dialograhmen

SK1, SK2 : Schulrechner-Kommunikationsabwicklungsinstanzen
 skz1, skz2 : Schulrechner-Kommunikationsabwicklungs-Zustandskanäle
 skü1, skü21, skü22 : Schulrechner-Kommunikationsabwicklungs-Übergabekanäle

GK1, GK2, gkz1, gkz2, gkü1, gkü21, gkü22: entsprechend für den Großrechner)

Eine von einer Verarbeitungsinstantz geforderte Ausgabe an den Benutzer wird entsprechend dem Benutzertyp durch die Kommunikationsausgabeinstanz AK3 (vgl. Abb. 13.4.) konkretisiert und realisiert, bevor sie an die Nachfolgeinstanz weitergegeben wird. Entsprechend wird umgekehrt die Kommunikationseingabeinstanz AK1 (vgl. Abb. 13.4.) Eingaben vom Benutzer entgegennehmen.

Wegen der geforderten Anwendungsunabhängigkeit des allgemeinen Dialograhmens müssen in diesem Bereich gleichermaßen Anpassungen von Funktionen an das (bzw. des) jeweilige(n) Anwendungsprogramm(s) vorgenommen werden. Diese erfolgen im Falle von Funktionen, die sich auf den Arbeitsplatzrechner beziehen durch die Instanzen AK2 und AK4. Innerhalb des Schulrechners und des Großrechners sind für diese Aufgaben die Instanzen SK1 und SK2, bzw. GK1 und GK2 vorgesehen. Diese Trennung bei den Kommunikationsinstanzen in erstens solche, die für die Geräteunabhängigkeit sorgen, zweitens in einen Bereich, der geräte- und anwendungsunabhängig operiert und drittens in Instanzen als Verbindung zu Anwendungsprogrammen ist vergleichbar der Definition einer "E/A-Schicht" und einer "Dialog-Schicht" von Balzert. (/Balzert86/).

Eine letzte Verfeinerung der Kommunikationsinstanz des Arbeitsplatzrechners betrifft die Manipulation von Fenstern und Darstellungsfeldern. Durch zwei Instanzen AKWE und AKWA stellen wir innerhalb der Benutzerschnittstelle spezielle "Window-Manager" dar, die die Aufgaben, die über die von GKS zur Verfügung gestellten Fähigkeiten hinausgehen, erledigen.

(AKWE: Arbeitsplatzrechner-Kommunikationsabwicklung-Window-Manager für Benutzereingaben

AKWA: Arbeitsplatzrechner-Kommunikationsabwicklung-Window-Manager für Benutzerausgaben

akzw: Arbeitsplatzrechner-Kommunikationsabwicklungs-Zustandskanal für die Window-Manager)

Hierzu zählt die allgemeine Verwaltung sämtlicher dem Benutzer in einem Arbeitszusammenhang zur Verfügung stehenden Fenster genauso wie die Cursorsteuerung: Ein Benutzer kann zu einem beliebigen Zeitpunkt durch ein Eingabegerät (z. B. die Maus) den Cursor auf dem Bildschirm bewegen. Je nachdem, über welchem Fenster der Cursor gerade steht, kann u. a. die Darstellungsart des Cursors wechseln (senkrechter Strich, Pfeil, waagerechter Strich). Die "Eingabe des Benutzers" entspricht dem Bewegen des Stiftes über einem Blatt Papier bei herkömmlicher Arbeit und ist im Sinne der abstrakten Dialogformen keine eigenständige Form. Der Benutzer löst keine vom System zu bearbeitende Funktion aus. Aus diesem Grund werden derartige "Eingaben" nicht von der Verarbeitungsinstanz des Arbeitsplatzrechners bearbeitet, sondern eine Systemreaktion findet innerhalb der Benutzerschnittstelle und hier durch die neugeschaffene Instanz AKDA als direkte Ausgabe statt.

(AKDA: Arbeitsplatzrechner-Kommunikationsabwicklungs-Instanz für direkte Ausgaben)

Betrachten wir den Nachrichtenfluß innerhalb des Modells aus Abb. 13.4. für einige Beispiele wieder genauer.

Eine **direkte Ausgabe** wird von der Benutzerschnittstelle erzeugt, wenn eine Benutzereingabe keine syntaktischen oder semantischen Veränderungen der Ausgabeinformation bewirkt und sich nur auf die Präsentation der Fenster oder des Cursors bezieht. Hierbei wird die Eingabe-Kombination mü1 - MKE - man - AKE durchlaufen und die Nachricht bei akü11 abgelegt. Die Instanz AKGE nimmt die Nachricht auf, wandelt sie in dialograhmenspezifische Form um und legt sie bei akü12 ab. Daraufhin verarbeitet die Window-Eingangsinstanz AKWE die Nachricht und stellt fest, daß keine Funktion vom Benutzer ausgelöst wurde. AKWE setzt daraufhin AKDA als Nachfolgeinstanz fest, vermerkt die Fenster- und Cursoraten bei akzw und übergibt die Nachricht, daß eine direkte Ausgabe erforderlich ist, an akü13. Diese Nachricht wird von AKDA aufgenommen und an akü23 weitergeleitet. Die Instanz AKWA ermittelt dann die dialograhmenspezifischen Ausgabedaten mithilfe von akwz und übergibt sie an AKGA. Hier werden sie in die gerätespezifische Form überführt und über die "Ausgabe-Pipeline" akü21 - AKA - amn - MKA - mü2 zur menschlichen Verarbeitungsinstantz geleitet.

Eine vom Benutzer ausgelöste **Funktion innerhalb des normalen Dialogs** wird über die "Eingabe-Pipeline" nach akü11 überführt, dort von AKGE aufgenommen und in dialograhmenspezifischer Form bei akü12 abgelegt. Die Instanz AKWE übergibt sie nach der Feststellung, daß keine direkte Ausgabe ausgelöst wurde, mit der Festlegung von AK1 als Nachfolger an den Kanal akü13. Hier wird sie von AK1 aufgenommen und analysiert.

Betrifft die Funktion den Arbeitsplatzrechner, so wird die Nachricht mit entsprechender Nachfolgemarkierung an akü24 übergeben und von AK4 aufgenommen. Diese Instanz legt die Verarbeitungsinstantz des Arbeitsplatzrechners AN als Nachfolger fest, wandelt die Nachricht in anwendungsabhängige Form um und stellt sie akü25 zur weiteren Verarbeitung zur Verfügung. Nach Aufnahme und Bearbeitung der Funktion durch AN wird eine Systemreaktion bei akü14 abgelegt. AK2 nimmt die Nachricht auf, stellt sie in anwendungsunabhängiger Form dar, legt die Benutzerausgabeinstanz AK3 als Nachfolger fest und übergibt die Nachricht an den Kanal akü24. Nach Aufarbeitung entsprechend der Konkretisierungs- und Realisierungsform des aktuellen Dialogs durch AK3 gelangt die Nachricht dann wieder über akü23 - AKWA - akü22 - AKGA - akü21 - AKA - amn - MKA - mü2 an die Verarbeitungsinstantz des Benutzers.

Betrifft eine von AK1 analysierte Funktion eine nachgelagerte Rechnerkomponente, so ist gegebenenfalls der Dialog mit dem Schulrechner zu eröffnen, bevor die Funktion dort verarbeitet werden kann. Dies erfolgt in der im letzten Abschnitt beschriebenen Weise, wobei wiederum akz1 und az Markierungen enthalten, die Auskunft darüber geben, ob ein aktueller Dialog mit einer nachgelagerten Systemkomponente besteht oder nicht.

Auch für die **Dialogbeendigung** mit dem Schulrechner gilt das im letzten Abschnitt Gesagte, wobei anstelle der Kommunikationsinstanzen SK (bzw. GK beim Großrechner) für die eine Richtung des Nachrichtenflusses die Instanzen SK1 (bzw. GK1) und für die andere SK2 (bzw. GK2) zuständig sind.

13.3. Ein drittes Systemmodell für den allgemeinen Dialograhmen mit der Dialogform Unterbrechung

Ein wesentlicher Aspekt der Anforderungen an einen allgemeinen Dialograhmen aus Teil II der Arbeit bestand in der Berücksichtigung unterschiedlicher Dialogformen eines Mensch-Rechner-Dialogs. Datenabfrage, Auswahlangebot, Kommando und Unterbrechung sind danach in benutzeradäquater Form zu einer abstrakten Dialogstruktur zu kombinieren, um das Systemverhalten benutzerfreundlich zu gestalten (vgl. Kapitel 7).

Bisher ist die abstrakte Dialogstruktur in den Modellen für den allgemeinen Dialograhmen nicht explizit berücksichtigt, und es ist denkbar, diesen Aspekt auf dieser Ebene auch nicht weiter zu betrachten, sondern ihn als Teil der Realisierung zu verstehen. Dann ist die abstrakte Dialogstruktur Bestandteil der die Kommunikationsabwicklungsinstanzen realisierenden Algorithmen und somit implizit im Modell enthalten. Allerdings verschleiert diese Interpretation eine ganze Reihe von Schwierigkeiten, die bei der Berücksichtigung der Dialogform "Unterbrechung" zu beachten sind: Wird durch eine Unterbrechung eine gerade aktuelle Arbeitsumgebung verlassen, so sind Aktionen innerhalb des in der Verarbeitung unterbrochenen Rechners durchzuführen, um bisherige Arbeitsergebnisse zu sichern, die Unterbrechungsstelle für eine spätere Wiederaufnahme der Arbeit zu speichern und den Rechner für eine neue Funktion vorzubereiten.

Um diese Aktionen im Modell nachbilden zu können, muß die abstrakte Dialogstruktur explizit im Petri-Netz dargestellt werden. Eine Möglichkeit, wie dies geschehen kann, beschreibt Saathoff /Saathoff82/, die in ihren Dialogmodellen sogenannte Abbruch- und Einschubaufträge durch Instanzen-Kanal-Beziehungen berücksichtigt. Ihre Darstellungen beziehen sich dabei auf einfache Systeme mit einem Benutzer und einem Rechner und können deswegen ohne Modifizierungen nicht auf den allgemeinen Dialograhmen übertragen werden.

Ausgangspunkt der Überlegungen von Saathoff ist eine "Unterbrechungsinstanz" beim Menschen mit direkter Verbindung zu den den Rechner darstellenden Instanzen und eine Aufteilung eines Auftrages in n Teilschritte mit der Definition je einer Bearbeitungsinstanz des Rechners für jeden Teilschritt. Bei einer Unterbrechung durch Schalten der menschlichen Unterbrechungsinstanz wird die Auftragsabwicklung innerhalb des Rechners nach einem der n Teilschritte

gestoppt. (Vgl. Abb. 13.5.: Dem rechnerseitigen Unterbrechungskanal wird durch Schalten der "menschlichen" Unterbrechungsinstanz die Marke entzogen. Dadurch kann keine Auftragsbearbeitungsinstanz mehr schalten, d.h. den Auftrag weiter bearbeiten. Eine Systemreaktion auf die Unterbrechung wird durch die rechnerseitige Unterbrechungsinstanz erzeugt und über den normalen Ausgabeweg an den Benutzer geleitet.)

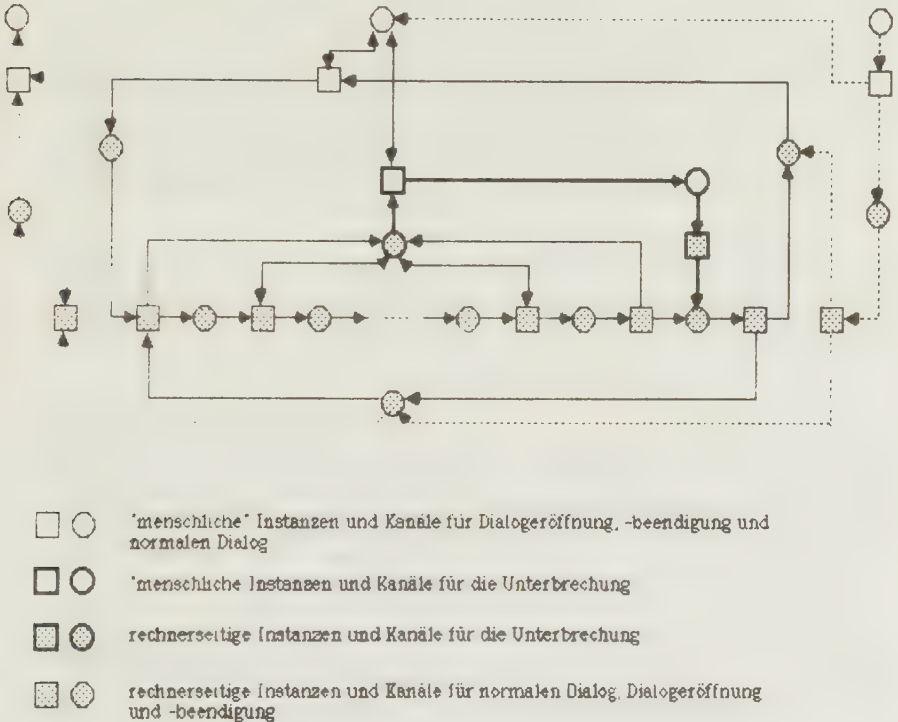


Abb. 13.5: Dialogmodell mit Unterbrechungsmöglichkeit (nach /Saathoff82/, S.116)

In Weiterentwicklungen des Modells aus Abb. 13.5 wird zusätzlich die das Ergebnis der Auftragsbearbeitung bis zu diesem Zeitpunkt darstellende Marke bei Unterbrechung gespeichert, sodaß ein späteres Wiederaufsetzen an der Abbruchstelle möglich wird. Mehrfache Unterbrechungen werden durch eine

Kellerstruktur des Speichers berücksichtigt (vgl. /Saathoff82/, S. 111-143). Saathoff stellt in ihren Modellen sämtliche notwendigen Aktionen durch Instanzen und Kanäle dar, verwendet hier also keine Attributsmarken oder Instanzenvorschriften. Sie erhält schon für einfache Dialoge und Systeme ausgesprochen komplexe Petri-Netze.

Aus zwei Gründen erscheint es für unseren Zweck wenig sinnvoll, den von Saathoff gewählten Weg weiterzuführen und auf den allgemeinen Dialograhmen zu übertragen: Zum einen wären die entstehenden Petri-Netze anschaulich nicht mehr darstellbar (vgl. die Abbildungen 25 - 30 in /Saathoff82/, S. 140-143), zum zweiten steht ihre Vorgehensweise in einem Punkt prinzipiell im Widerspruch zu den Grundsätzen des allgemeinen Dialograhmens. Die Modelle von Saathoff gehen von einer direkten Unterbrechungsverbindung vom Menschen zum Rechner aus, was übertragen auf den allgemeinen Dialograhmen eine direkte Verbindung vom Menschen zum Schul- und Großrechner erfordern würde. Diese ist von der Architektur des allgemeinen Dialograhmens her aber ausdrücklich ausgeschlossen.

Wir wollen deshalb zur Nachbildung der abstrakten Dialogstruktur einen Mittelweg zwischen den beiden genannten Alternativen gehen und notwendige Aktionen bei Dialogformen teilweise durch Attributsmarken und Instanzenvorschriften (also realisierungsnah), teilweise aber auch durch Definition spezieller Instanzen und Kanäle berücksichtigen.

Ausgangspunkt unserer Modellierung ist - wie bei Saathoff - die Bearbeitung eines Auftrags durch den Rechner in n Teilschritten mittels je einer Instanz (also die Verfeinerung der Instanzen AN, SN und GN aus Abb. 13.4), sowie eine "menschliche" Unterbrechungsinstanz, die allerdings über die normale Eingabesequenz von Kanälen und Instanzen mit den Rechnern verbunden ist. Daneben gibt es jeweils in den Rechnern Instanzen zum Speichern von Abbruchstellen und Wiederaufsetzen an solchen Abbruchstellen.

Das Petri-Netz für dieses dritte Systemmodell für den allgemeinen Dialograhmen ist in Abb. 13.6 dargestellt. Als Bezeichnungen wurden gewählt (neben den in den vorherigen Abschnitten beschriebenen):

MU:	menschliche Unterbrechungsinstanz
manu:	Mensch-Arbeitsplatzrechner-Kanal für normalen und Unterbrechungsdialog
mu1,mu2:	menschliche Unterbrechungszustandskanäle
AKAU:	Arbeitsplatzrechner-Kommunikationsinstanz für Ausgabeunterbrechungen
AKGU:	Arbeitsplatzrechner-Kommunikationsinstanz für GKS-Unterbrechungen
AKWU:	Arbeitsplatzrechner-Kommunikationsinstanz für

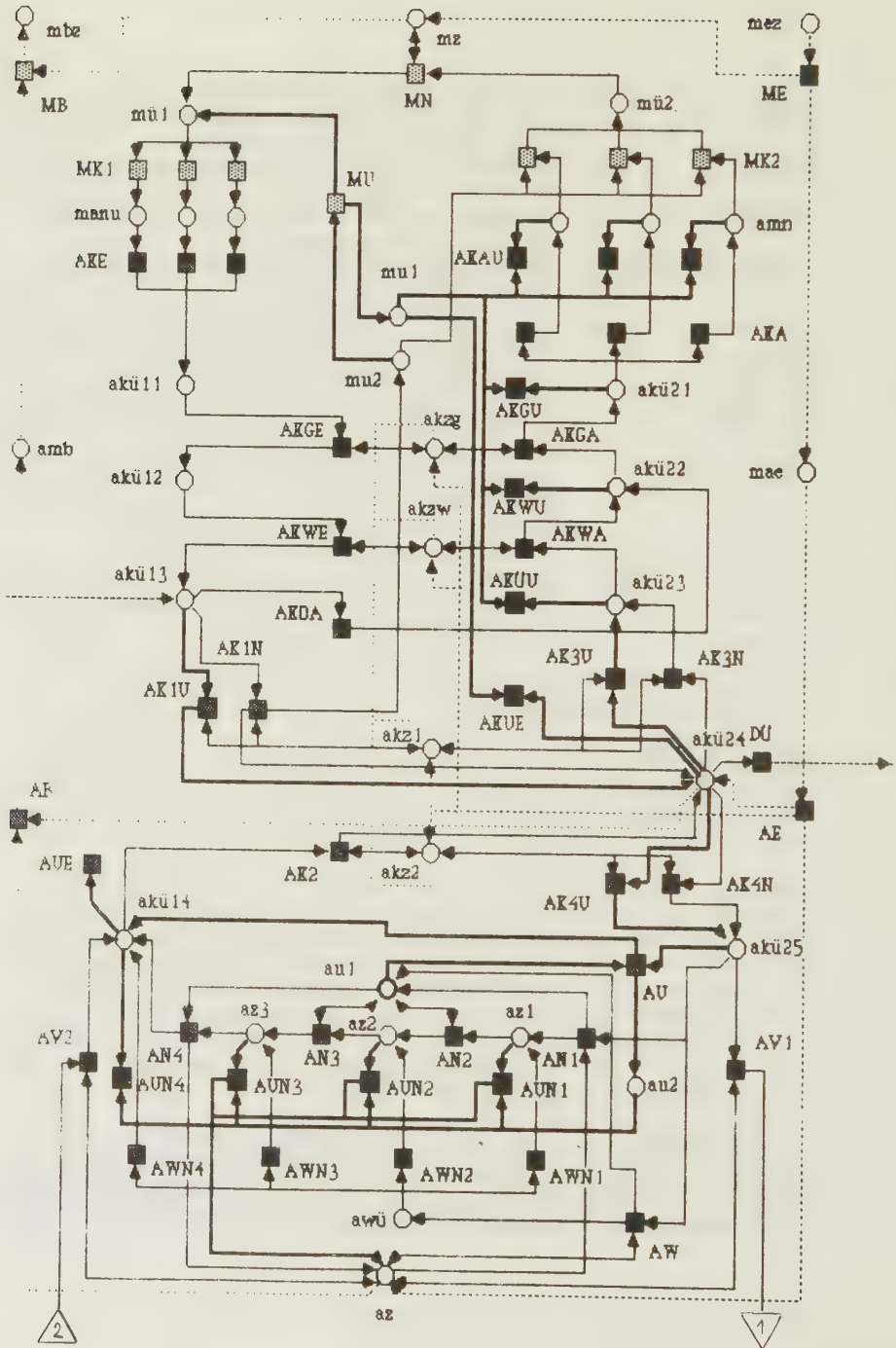
	Window-Manager-Unterbrechungen
AKÜÜ1:	Arbeitsplatzrechner-Kommunikationsinstanz für Übertragungs-Unterbrechungen
AKUE:	Arbeitsplatzrechner-Kommunikationsinstanz für Unterbrechungs-Ende
AK1U, AK3U, AK4U:	Arbeitsplatzrechner-Kommunikationsinstanzen für Unterbrechungsabwicklung
AK1N, AK3N, AK4N:	Arbeitsplatzrechner-Kommunikationsinstanzen für normale Dialogabwicklung
AU:	Arbeitsplatzrechner-Unterbrechungsinstanz
AUE:	Arbeitsplatzrechner-Unterbrechungs-Ende_Instance
au1, au2:	Arbeitsplatzrechner-Unterbrechungskanäle
AN1 - AN4:	Arbeitsplatzrechner-Instanzen für normalen Dialog
AUN1 - AUN4:	Arbeitsplatzrechner-Instanzen für Unterbrechungen des normalen Dialogs
AWN1 - AWN3:	Arbeitsplatzrechner-Instanzen für Wiederaufnahme des normalen Dialogs
AW:	Arbeitsplatzrechner-Wiederaufnahme-Instanz
awÜ:	Arbeitsplatzrechner-Wiederaufnahme-Übertragungskanal
az1 - az3:	Arbeitsplatzrechner-Zustandskanäle für normale Dialogschritte

Die übrigen Bezeichnungen folgen mit "s" für Schulrechner und "G" für Großrechner der obigen Systematik.

Im Modell aus Abb. 13.6 ist aus Gründen der Übersichtlichkeit eine Aufteilung für die Bearbeitung eines Auftrages in nur vier Teilschritten dargestellt. Weiter wurden alle Relationen und Instanzen zur Berücksichtigung von Aktivitäten im Zusammenhang mit der Unterbrechung fett gedruckt.

Betrachten wir anhand des Modells den Nachrichtenfluß für folgenden Beispiel-Dialog im Detail: Der Benutzer beginnt seine Arbeit am Rechner und führt zunächst einen normalen Dialog mit dem Arbeitsplatzrechner. Zu seiner Information bedient er sich dann durch Aufruf einer entsprechenden Funktion eines Datenbanksystems des Großrechners und wechselt danach wieder in den normalen Dialog mit dem Arbeitsplatzrechner. Er unterbricht diesen, um eine Standardanwendung, z.B. den Taschenrechner, des Arbeitsplatzrechners zu benutzen. Nach Wiederaufnahme des normalen Dialogs beendet er nach einigen Dialogschritten seine Arbeit am Rechner.

Die **Dialogeröffnung mit dem Arbeitsplatzrechner** verläuft im wesentlichen, wie bereits in Abschnitt 13.1. beschrieben: Das Schalten von ME bewirkt bei mae eine Markierung, daß der Dialog mit dem Arbeitsplatzrechner eröffnet werden soll. Diese Marke nimmt die Instanz AE beim Schalten auf, um



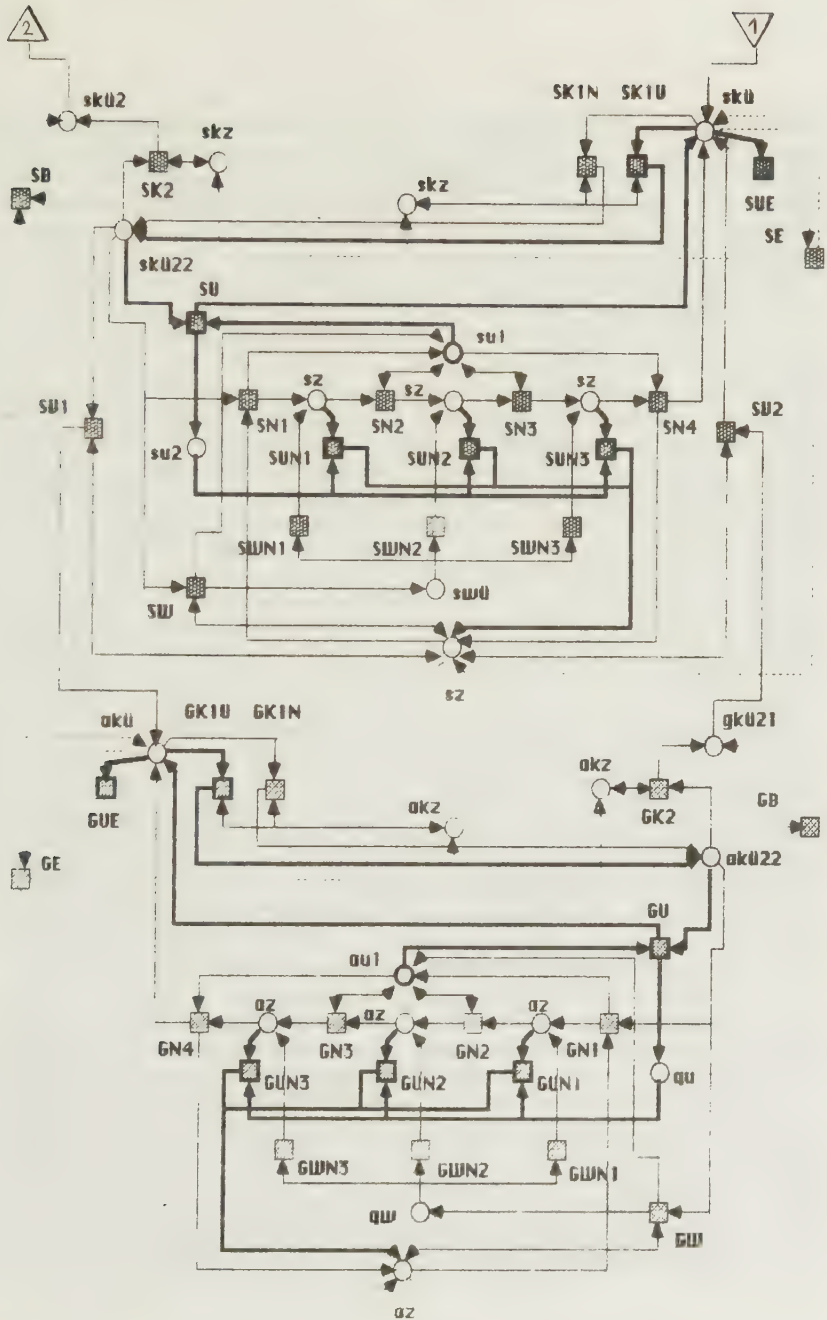


Abb. 13.6: Das "Dritte Systemmodell" für den allgemeinen Dialographen

durch Markierung und Attributierung sämtlicher Zustandskanäle (az, akz1, akz2, akzw, akzg) den Arbeitsplatzrechner betriebsbereit zu machen, d.h. Dateien zu öffnen, Programme zu starten, Voreinstellungen zu setzen. Weiter legt AE eine Marke bei dem Kanal akü24 zur Vorbereitung einer Ausgabe ab und bestimmt durch Attributierung AK3N als Nachfolgeinstanz.

Mit Hilfe des Zustandskanals akz1, den dort vorliegenden Voreinstellungen und der Markierung von akü24 bestimmt AK3N Form und Inhalt der Ausgabe, legt dabei insbesondere die Dialogform, -konkretisierung und -realisierung für den nächsten Dialogschritt fest und übergibt die Nachricht mit Markierung und Attributierung an den Kanal akü23. AKWA ist jetzt schaltfähig, legt auf der Grundlage von akzw die Fensterstruktur fest - komplettiert dadurch die Dialogrealisierung - und übergibt die Nachricht als Marke an den Kanal akü22. Daraufhin wird sie von AKGA bearbeitet und in entsprechend aufbereiteter Form an akü21 übergeben. Durch Attributierung wird dabei das durch die Realisierung vorgesehene Ausgabegerät festgelegt, d.h. eine Instanz AKA als Nachfolger bestimmt. Über AKA, amn, MK2 und mü2 gelangt die Nachricht an die menschliche Verarbeitungsinstanz MN.

Der Benutzer beginnt einen **normalen Dialog mit dem Arbeitsplatzrechner**, indem er entsprechend der abstrakten Dialogstruktur eine Systemfunktion in der ihm zur Verfügung gestellten Dialogform auslöst. Im Modell entspricht dies dem Schalten der Instanz MN und Markieren und Attributieren von mü1. Die dort dadurch vorliegende Nachricht wird zuerst von AKGE und AKWE bearbeitet, analysiert und in dialograhmenspezifische Form gebracht. Entsprechend der Dialogform ist die Instanz AK1N Nachfolger. Die Aufgabe dieser Instanz besteht in der Feststellung der vom Benutzer ausgelösten Funktion und damit auch der Bestimmung der diese Funktion bearbeitenden Nachfolgeinstanz (AN1 im Falle einer an den Arbeitsplatzrechner gerichteten Funktion, DÜ für Nachrichten an andere Benutzer, AV1 sonst.). Dazu bedient sich die Instanz ihres Zustandskanals akz1. Weiter wird die Benutzernachricht hier auf Zulässigkeit untersucht und gegebenenfalls eine Fehlermeldung generiert. Mit diesen Informationen wird die Nachricht dann an den Kanal akü24 übergeben.

Als Nachfolgeinstanz kommen AK4N, DÜ oder für Ausgaben an den Benutzer AK3N in Frage. Für unseren Beispieldialog entscheidet AK4N den Konflikt für sich, wandelt die Nachricht in anwendungsspezifische Form um und übergibt sie an akü25 mit der Festlegung von AN1 als Nachfolgeinstanz. Die vom Benutzer ausgelöste Funktion wird in Teilschritten durch AN1, AN2, AN3 und AN4 bearbeitet, wobei die Voraussetzungen zur Bearbeitung durch den Zustandskanal az gegeben und Zwischenergebnisse bei az1, az2 und az3 abgelegt werden. Eine Sonderrolle übernimmt dabei der Kanal au1. Eine Markierung auf diesem Kanal zeigt an, daß zur Zeit keine Unterbrechung erfolgt ist, die normale Bearbeitung also fortgesetzt werden kann.

Nach Bearbeitung der Funktion wird das Ergebnis als Nachricht bei akü14 gespeichert, von wo aus durch AK2 eine Ausgabe an den Benutzer eingeleitet wird. AK2 transformiert dazu die anwendungsspezifische Nachricht in dialograhmenspezifische Form, übergibt sie an akü24 und legt AK3N als Nachfolgeinstanz fest. Hier wird mit Hilfe von akz1 Dialogform, -konkretisierung und -realisierung für die Ausgabe bestimmt und die Nachricht zur weiteren Bearbeitung an akü23 übergeben. Über die Instanzen AKWA, AKGA, AKA und MK2 gelangt sie dann zur menschlichen Verarbeitungsinstanz MN. Der geschilderte Ablauf beginnt von Neuem.

Wenn - entsprechend unserem vorausgesetzten Beispiel-Dialog - die Instanz AK1N bei der Analyse der Benutzernachricht nicht den Arbeitsplatzrechner als Ansprechpartner findet, so muß - außer im Falle von DÜ - **der Dialog mit dem Schulrechner eröffnet werden**. Dazu wird zunächst bei akz1 eine entsprechende Marke gesetzt und dann AV1 als Ziel-Instanz festgelegt. Nach Weiterleitung der Nachricht durch AK4N kann AV1 die Nachricht vom Kanal akü25 aufnehmen. Durch Markierung von az "weiß" AV1, daß, bevor der Schulrechner die Nachricht bearbeiten kann, zunächst erst der Dialog mit dieser Systemkomponente eröffnet werden muß. AV1 legt dazu die Nachricht mit der Festlegung von SE als Nachfolgeinstanz bei skü1 ab. SE macht daraufhin den Schulrechner betriebsbereit (Markierung und Attributierung von sz, skz1 und skz2) und legt SK1N als Nachfolger fest.

SK1N stellt mit Hilfe von skz1 fest, daß der Schulrechner nicht der gewünschte Ansprechpartner ist und daß **der Dialog mit dem Großrechner zu eröffnen ist**. Dies vermerkt sie bei skz1 und zusammen mit der Benutzernachricht auch durch Markierung und Attributierung bei skü22. SV1 ist dann Nachfolgeinstanz und bestimmt zur Dialogeröffnung mit dem Großrechner GE als Nachfolger. Diese markiert - wie vorher SE - die zugehörigen Zustandskanäle und beläßt die Nachricht zur Weiterverarbeitung durch GK1N auf gkü1. Die Instanz analysiert die vom Benutzer ausgelöste Funktion, wandelt sie in anwendungsabhängige Form um und legt die Instanz GN1 als Nachfolger fest.

Die Bearbeitung der vom Benutzer gestellten Aufgabe beginnt durch Schalten von GN1 und wird durch GN2, GN3 und GN4 fortgesetzt. GN4 generiert zum Bearbeitungsende eine an den Benutzer gerichtete Ausgabe und legt sie mit der Bestimmung von GK1N als Nachfolger bei gkü1 ab. Diese Instanz findet daraufhin GK2 als Nachfolger und legt die Ausgabenachricht bei gkü22 ab. GK2 transformiert die Information in dialograhmenspezifische Form, woraufhin SV2 die Nachricht aufnehmen kann und mit der Festlegung von SK1N als Nachfolger an skü1 übergeben kann. Diese Instanz bestimmt SK2 als Nachfolger und legt die Ausgabenachricht bei skü22 ab. Nach Schalten von SK2, AV2 und AK2 gelangt die Nachricht dann nach akü14, von wo sie in gewohnter Weise an die Benutzerinstanz MN weitergeleitet wird.

Nehmen wir an, daß dessen Aktion bzw. Reaktion wiederum an den Großrechner gerichtet ist, so wird nach Durchlaufen der Arbeitsplatzrechner-Kommunikationsinstanzen AKE, AKGE und AKWE durch AK1N festgestellt, daß wiederum eine Funktion ausgelöst, bzw. Eingabe getätigt wurde, die nicht vom Arbeitsplatzrechner zu bearbeiten ist. Daraufhin wird die Instanz AV1 als Ziel festgelegt und die Benutzernachricht an akü24 übergeben. Nach Übergabe durch AK4N nimmt AV1 die Nachricht auf, stellt anhand von az fest, daß der Dialog mit dem Schulrechner eröffnet ist und übergibt mit der Festlegung von SK1N als Nachfolger die Nachricht an skü1. Hier wird wieder festgestellt, daß nicht der Schulrechner durch den Benutzer angesprochen wurde und demnach SV1 Nachfolgeinstanz ist. Da der Dialog mit dem Großrechner eröffnet ist, kann nach Schalten von SV1 die Instanz GK1N die Benutzereingabe analysieren, transformieren und die Instanz GN1 zum Nachfolger bestimmen. Hier wird die Bearbeitung der Eingabe begonnen, durch GN2 und GN3 fortgesetzt und GN4 letztlich abgeschlossen. Wieder wird eine Ausgabe an den Benutzer generiert und auf die oben beschriebene Weise nach MN geleitet.

Der Benutzer reagiert jetzt durch Auslösen einer den Arbeitsplatzrechner betreffenden Funktion. Dies wird nach den beschriebenen vorausgehenden Transformationen durch AK1N festgestellt. Da diese Instanz anhand der Markierung von akz1 findet, daß zum gegenwärtigen Zeitpunkt noch ein Dialog mit dem Schulrechner besteht, leitet sie die Dialogbeendigung mit dieser Systemkomponente ein. Sie vermerkt dies bei akz1 und legt mit dem Vermerk, daß AK4N die vom Benutzer ausgegangene Nachricht bei akz4 zwischenspeichern muß, AV1 als Zielinstanz fest. Nachdem über AK4N und AV1 die Beendigungsnachricht bei skü1 angelangt ist, ermittelt die Instanz SK1N mithilfe von skz1, ob zum gegenwärtigen Zeitpunkt noch ein Dialog mit dem Großrechner besteht. Dies ist für unser Beispiel der Fall.

Daraufhin wird die Beendigungsnachricht für den Schulrechner mit dem Vermerk, daß der Dialog mit dem Großrechner beendet wird, bei skz1 gespeichert und die Beendigungsnachricht für den Großrechner mit der Festlegung von SV1 als Nachfolger nach skü22 übertragen. SV1 legt dann GB als Nachfolgeinstanz fest und übergibt eine entsprechende Marke an gkü1. Das darauffolgende Schalten von GB führt zum Abzug der Zustandsmarken auf gkz1, gkz2 und gz, sowie der Marke auf gkü1 und zur Generierung einer Marke auf gkü21 mit der Nachricht, daß der Dialog mit dem Großrechner nicht mehr besteht.

Diese Nachricht gelangt zu SV2, die diese Tatsache bei sz speichert und die Nachricht an skü1 weitergibt. Dort wird sie von SK1N aufgenommen; mit Hilfe von skz1 wird festgestellt, daß die nächste auszuführende Funktion die Dialogbeendigung mit dem Schulrechner ist. Es wird somit diese Nachricht an skü22 übergeben, wo sie dann SB als Nachfolgeinstanz erreicht. SB zieht daraufhin die Marken von den Zustandskanälen des Schulrechners ab und legt eine Markierung über das Dialogende auf den Kanal skü21. AV2 vermerkt die

Dialogbeendigung bei az und gibt eine entsprechende Nachricht an AK2 weiter. Die Instanz AK2 legt jetzt AK4N als Nachfolger fest, damit von dort die ursprünglich vom Benutzer ausgegangene Nachricht durch den Arbeitsplatzrechner bearbeitet werden kann. Diese Bearbeitung erfolgt auf die oben geschilderte Weise als normaler Dialog mit dem Arbeitsplatzrechner.

Nach einigen Dialogschritten folgt laut unserem Beispiel eine **Unterbrechung des normalen Dialogs**. Jede Unterbrechung wird initiiert durch Schalten der "menschlichen" Instanz MU. Gemäß dem Modell aus Abb. 13.6 ist diese nur schaltfähig, wenn der Kanal mu2 markiert ist. Da dessen Markierung durch die Instanz AK1N vorgenommen wird, kann der Dialog also erst dann unterbrochen werden, wenn ein vorausgegangener Benutzerauftrag diese Instanzen bereits durchlaufen hat, der Arbeitsplatzrechner (oder eine nachgelagerte Systemkomponente) also eine Funktion zur Bearbeitung vorliegen hat. Diese Voraussetzung ist sinnvoll und notwendig, um im eigentlichen Sinn von "Unterbrechen" sprechen zu können und weiter ein in sich konsistentes Modell für eine Unterbrechung zu bilden.

Wir gehen als Grundlage für die Unterbrechung eines normalen Dialogs außer von der oben genannten Einschränkung weiter von folgender Fallunterscheidung aus:

- Die Unterbrechung erfolgt zu einem Zeitpunkt, an dem eine vorherige Aufgabe durch den Arbeitsplatzrechner (oder einen nachgelagerten Rechner) direkt bearbeitet wird und somit eine der Instanzen AN1 bis AN4 (oder die entsprechenden nachgelagerten) aktiv ist. Dann entstehen durch die Unterbrechung Zwischenergebnisse, die gespeichert werden müssen.
- Die Unterbrechung erfolgt zu einem Zeitpunkt, zu dem die eigentlichen Bearbeitungsinstanzen den zu unterbrechenden Auftrag bereits erledigt haben, und durch AN4 eine Ausgabe an den Benutzer generiert wurde. Jetzt ist der Ausgabevorgang zu unterbrechen, wobei wir zur Vereinfachung des Modells davon ausgehen, daß dann der Unterbrechungszeitpunkt und die an den Benutzer gesandte Nachricht nicht gespeichert werden. Bei Wiederaufnahme der Bearbeitung des unterbrochenen Auftrages muß diese daher durch AWN4 unter Zuhilfenahme von az neu generiert werden.

Im Detail stellt sich der Nachrichtenfluß für eine Unterbrechung des Arbeitsplatzrechners dann wie folgt dar (Unterbrechungen des Schul- oder Großrechners werden mit den dort definierten Instanzen entsprechend durchgeführt): Das Schalten von MU führt zur Markierung der Kanäle mü1 und mu1. Von dem ersteren gelangt die Unterbrechungsnachricht über die normalen Eingabeinstanzen zur Unterbrechungseingabeinstanz des Arbeitsplatzrechners AK1U und von dort zum zentralen Verteil-Kanal akü24. (Die Marke auf mu1 dient dem Unterbrechen von Ausgaben (s.u.).)

Wurde eine Ausgabenachricht bereits mittels akü24 an den Benutzer weitergeleitet, so befindet sie sich als Markierung auf einem der Kanäle akü23, akü22, akü21 oder amn. Da der Kanal mu1 markiert ist - als Zeichen, daß eine Unterbrechung aktuell ist -, ist somit eine der Instanzen AKÜÜ, AKWU, AKGU oder AKAU schaltfähig, schaltet auch und zieht damit die Marken auf ihren beiden Eingangskanälen ab. Dies ist zu interpretieren als Vernichtung der Ausgabenachricht. Ist für den vorangegangenen Dialogschritt noch keine Ausgabe erfolgt, so ist die Instanz AK4U Nachfolger. Sie überführt - wie bei "normalen" Dialogschritten AK4N - Nachrichten von der dialograhmenspezifischen in anwendungsabhängige Form und bestimmt die Unterbrechungsinstanz AU des Arbeitsplatzrechners als Nachfolger. AU seinerseits beendet die Bearbeitung des vorangegangenen Auftrags, indem die auf au1 befindliche Marke abgezogen wird. Da jetzt keine der Instanzen AN1 bis AN4 oder AK2 mehr schalten kann, ist vom Modell her die Bearbeitung des Auftrags unterbrochen.

War zu diesem Zeitpunkt eine der Instanzen AN1 bis AN4 aktiv, so befindet sich auf einem der Kanäle az1 bis az3 oder akü14 noch eine Marke, die das jeweils zuvor ermittelte Zwischenergebnis modelliert. Eine Speicherung dieser Information wird durch Schalten einer der Instanzen AUN1 bis AUN4 gewährleistet. Hier ist gerade diejenige Instanz schaltfähig, die sowohl auf au2 als auch auf az1, az2, az3 oder akü14 eine Marke findet. Das Schalten von AUN1 bis AUN4 führt zur Veränderung der Markierung von az, was als Speichern des Zwischenergebnisses zu interpretieren ist.

Wird gleichzeitig mit der Unterbrechung eine neue Funktion aktiviert (z.B. Wechseln zwischen zwei auf dem Bildschirm dargestellten und zu unterschiedlichen Anwendungen gehörenden Fenstern), so wird diese neue Funktion zunächst bei akz2 zwischengespeichert, bis die Unterbrechung abgeschlossen ist. Ist keine neue Funktion initialisiert, wird durch AK3U eine Unterbrechungsnachricht erzeugt. Unserem Beispiel zufolge wird nach der Unterbrechung durch den Benutzer ein neuer Anwendungsdialog begonnen. Dieser verläuft im Modell als normaler Dialog, wie oben beschrieben.

Nach Beendigung dieses zwischengeschobenen Anwendungsdialogs muß die Arbeit mit dem zuvor unterbrochenen Auftrag wieder aufgenommen werden. Dies geschieht im Modell mit Hilfe der Instanz AW. Dazu wird zunächst bei einer Benutzereingabe festgestellt, daß sich ein Dialogschritt auf einen früheren Arbeitszusammenhang bezieht; daraufhin wird die Instanz AW für die Bearbeitung vorgesehen. AW holt sich vom Zustandskanal az die benötigte Information zu dem wieder aufzunehmenden Auftrag und bestimmt eine der Instanzen AWN1 bis AWN4 als Nachfolger, die dann den jeweiligen Eingangskanal az1 bis az3 oder akü14 in der Form markieren, daß AN2 bis AN4 oder AK2 an der zuvor unterbrochenen Stelle weiterarbeiten können. Gleichzeitig markiert die Instanz AW den Kanal au1 als Zeichen, daß zur Zeit keine Unterbrechung vorliegt.

Nach Übergang zum normalen Dialog mit dem Arbeitsplatzrechner wird einige Dialogschritte später gemäß unserem Beispiel der Dialog mit dem Arbeitsplatzrechner beendet. Dieser Auftrag geht aus vom Benutzer - also der Instanz MN - und gelangt zu der Eingabeinterpretationsinstanz AK1N. Da dort vermerkt ist, daß mit einem nachgelagerten Rechner zur Zeit kein Dialog angemeldet ist, wird die Instanz AB als Nachfolger festgelegt und die Beendigungsnachricht nach akü24 übertragen. AB zieht daraufhin dort und von sämtlichen Zustandskanälen die Marken ab, schließt dadurch den Dialog mit dem Arbeitsplatzrechner ab und gibt eine entsprechende Nachricht an den Kanal amb. Das Schalten von MB beschließt dann den Dialog.

13.4. Bemerkungen zu den Systemmodellen

Es bleibt zum Schluß dieses Kapitels anzumerken, daß wir mit dem Modell aus Abb. 13.6 eine Darstellung für den allgemeinen Dialograhmen gefunden haben, die sowohl den geforderten Systemeigenschaften nach Geräte- und Anwendungsunabhängigkeit, nach Berücksichtigung der Klassifizierung nach dem Konkretisierungsgrad und damit nach Möglichkeiten zur Berücksichtigung des Aspekts Benutzerfreundlichkeit Rechnung trägt, als auch "einfach" und damit überschaubar ist. Wir haben damit das Ziel des Entwurfs praktikabler Modelle für den allgemeinen Dialograhmen als Grundlage für mögliche Realisierungen erreicht und somit die Ansätze von Saathoff /Saathoff82/ in wesentlichen Bereichen weiterentwickelt. Allerdings geht die Einfachheit zu Lasten der notwendigen Interpretation des Modells. Die Verschiebung diverser Einzelheiten von der Ebene der Instanzen und Kanäle auf die der Attributsmarken und Instanzenvorschriften führt dazu, daß das Modell allein mit der Schaltregel für markierte Petri-Netze nicht mehr ausreichend die gewünschten Systemeigenschaften wiedergibt, sondern zusätzliche Informationen durch Attributsmarken und Instanzenvorschriften notwendig sind.

Dem Modell aus Abb. 13.6 (und den vorausgegangenen) haben wir jeweils informelle Beschreibungen zur Interpretation der Darstellungen beigelegt, die als Vorbereitung der Entwicklung von Attributsmarken und Instanzenvorschriften zu verstehen sind. Diese Beschreibungen haben gezeigt, daß das Modell im gewünschten Sinne "funktioniert"; es wurde aber auch deutlich, daß das zuletzt entwickelte Netz nur eine Möglichkeit zur Modellierung wiedergibt. Zahlreiche andere Anordnungen von Instanzen und Kanälen sind denkbar und müssen im Verhältnis zur gewählten diskutiert werden, um ein optimales Netz für das gestellte Problem zu erhalten. Wir können dies im Rahmen dieser Arbeit nicht leisten. Insofern ist das entwickelte Netz in Verbindung mit den im folgenden Kapitel wiedergegebenen Algorithmen zur Darstellung eines allgemeinen Dialograhmens nur als erster Ansatz zur Lösung des Problems zu verstehen, der zukünftig weiterentwickelt und optimiert werden muß.

14. Attributsmarken und Instanzenvorschriften für das Systemmodell des allgemeinen Dialograhmens

Wir wollen die im letzten Kapitel informell geschilderte Modell-Funktionsweise durch Definition von Instanzenvorschriften und Attributsmarken formalisieren. Unsere Vorgehensweise lehnt sich dabei an die aus Abschnitt 11.4 an, allerdings mit einer grundlegenden Änderung: Die Attributierung von Übergabemarken wird dahingehend verändert, daß nicht mehr eine Form für alle Übergabekanäle, sondern je Kanal eine spezielle Attributierung gilt. Dadurch erreichen wir trotz großen Funktionenumfangs und vielfältiger Möglichkeiten des allgemeinen Dialograhmens übersichtliche und handhabbare Marken bei der Modellierung (vgl. auch die Bemerkungen in den Abschnitten 11.3 und 11.4, sowie die Definition der strukturierten, markierten Netze in Abschnitt 11.3).

14.1. Allgemeine Vereinbarungen

Durch das zuletzt entwickelte Systemmodell für den allgemeinen Dialograhmen (vgl. Abb. 13.6) wurde die Benutzerschnittstelle des Systems besonders hervorgehoben. Um dabei Anwendungs- und Geräteunabhängigkeit einerseits und Flexibilität des Dialogverhaltens ("jedem Benutzer seine Benutzerschnittstelle") andererseits zu gewährleisten, müssen die durch das System fließenden Nachrichten passend transformiert werden. Bezüglich dieser Transformationen - für die im Folgenden Marken (also Datenstrukturen) für Kanäle und Instanzenvorschriften anzugeben sind - können wir als Verfeinerung der Überlegungen des Kapitels 10 fünf Ebenen unterscheiden (vgl. Abb. 14.1).

Der **Benutzer** tätigt Eingaben, bearbeitet Ausgaben und verfolgt bei seiner Arbeit einen Zweck, hat Stimmungen, Gefühle usw. Wir wollen - wie in Abschnitt 11.3 - hier keine detaillierten Datentypen, Strukturen oder Algorithmen formulieren, sondern den Menschen als einen nicht näher zu differenzierenden "System-Bestandteil" betrachten, der - nach Starten des Systems - im allgemeinen ausgehend von einer Systemausgabe eine Eingabe tätigt.

Hierzu operiert er auf einer sogenannten **physikalischen Struktur** (des Mensch-Rechner-Dialogs), unter die wir Art und Anzahl der E/A-Geräte, die Form, wie Nachrichten präsentiert und Eingaben getätigt werden können, den aktuellen Arbeitszusammenhang usw. zusammenfassen wollen. Eine solche physikalische Struktur wird dem Benutzer vom System dargeboten, woraufhin dieser Handlungen plant, Entscheidungen trifft und Aktionen durchführt. Eine Aktion besteht in der Betätigung eines Eingabegerätes, d.h. dem Auslösen einer physikalischen Funktion und in der Übergabe einer Nachricht an das System.

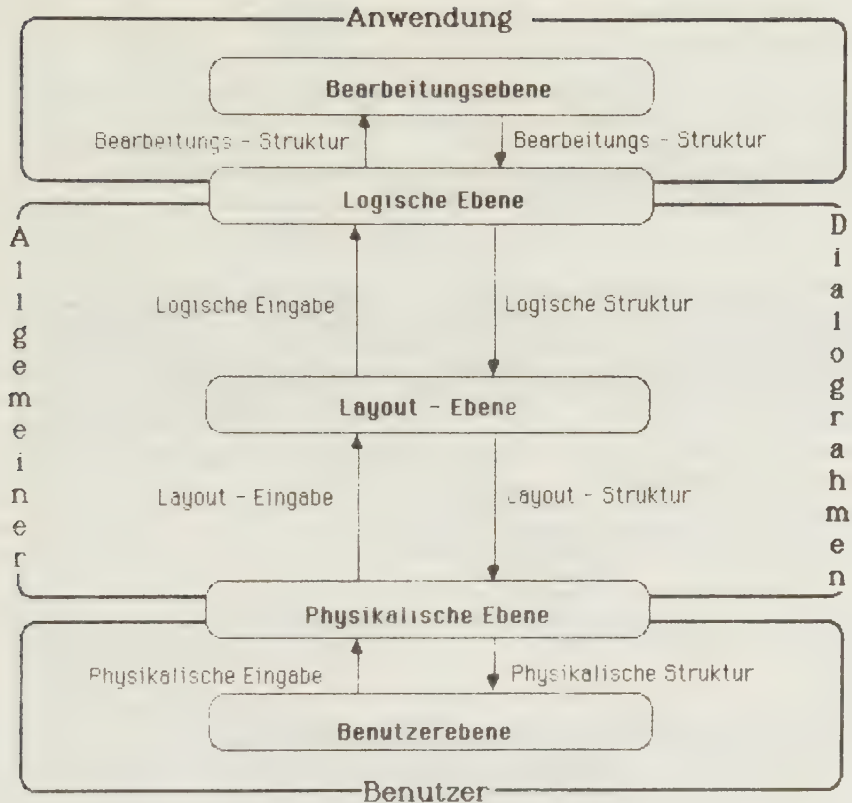


Abb. 14.1: Ebenen innerhalb des allgemeinen Dialograhmens

Zur formalen Beschreibung dieses Sachverhalts definieren wir zunächst einige grundlegende Datenstrukturen, aus denen dann die physikalische Struktur und die Benutzereingabe aufgebaut werden. Es ist:

Type Arbeitszusammenhang = (Kontrolle, Gestaltung, Meta, Anwendung1, ..., AnwendungN)

ein (unvollständiger) Aufzählungstyp, der die möglichen Arbeitsumgebungen innerhalb des Mensch-Rechner-Dialogs modelliert.

Type Physik_Funktionen = (.....)

ein (hier nicht näher spezifizierter) Aufzählungstyp zur Darstellung aller im System möglichen physikalischen Ein- und Ausgabefunktionen und

Type Dialogformen = (Unterbrechung, Kommando, Auswahlangebot, Datenabfrage);
 Abstrakte_Dialogstruktur = **Set of** Dialogformen

für die im Mensch-Rechner-Dialog möglichen Dialogformen.

Des weiteren sind für die physikalische Struktur die an den Benutzer übergebenen bzw. von ihm empfangenen Nachrichten, ihre Darstellungsform, sowie die beteiligten E/A-Geräte zu beachten. Hier gilt nach heutigem Erkenntnisstand für Bildschirmarbeit die Fenstertechnik als das grundlegende Gestaltungsmittel für den Nachrichtenaustausch. Wir haben diesem Umstand in dem Modell aus Abb. 13.6 dadurch Rechnung getragen, daß wir die Instanzen AKWE und AKWA insbesondere auch zur Fensterverwaltung eingeführt haben und definieren deshalb hier

Type Physik_Fensterstruktur = **Record**
 Gerät: Physik_Gerät;
 Nachricht: Text;
 Zuordnung: Arbeitszusammenhang;
 Realisierung: Physik_Realisierung
End

mit

Type Physik_Gerät = (Text_Bildschirm, Grafik_Bildschirm, Matrix_Drucker, Typenrad_Drucker, Sprachausgabegerät, Tongenerator, Modem, ..., Tastatur, Maus, Funktionstastatur, Digitalisierer, ...)

als (unvollständige) Aufzählung im Dialog tatsächlich vorhandener Geräte,

Type Text = **Array** [1.. Maxtextlänge] of Char

zur Darstellung von Nachrichten (Die Möglichkeit, zum Nachrichtenaustausch auch grafische Darstellungen zu benutzen, wird von uns zwar gesehen, doch wollen wir innerhalb des weiteren Vorgehens der Einfachheit halber Nachrichten als Texte modellieren. Eine Erweiterung an dieser Stelle bereitet auch keine grundsätzlichen Schwierigkeiten, wenn wir Verbunde zur Kombination von textlichen und grafischen Darstellungen verwenden.)

Type Physik_Realisierung = **Record**
 Schriftart : (...);
 Groß_und_Klein : (...);
 Farbe : (...);

```

Helligkeit : ( ... );
Blinken : ( ... );
Bewegung : ( ... );
....
End

```

wieder unvollständig als Aufzählungen möglicher Ausprägungen, bezogen auf die zur Verfügung stehenden Geräte. Bei der Belegung der Komponenten der physikalischen Fensterstruktur wird für das Gerät beim augenblicklichen Stand der Technik nur ein Bildschirm in Frage kommen.

Für Sprache und Ton z.B. kann sinnvoll nicht von einer Fensterstruktur ausgegangen werden. Wir sehen deshalb für die physikalische Struktur noch eine Komponente Physikalische Informationseinheiten vor, durch die die übrigen Ein- und Ausgaben berücksichtigt werden können. Mit den gleichen Komponenten, wie die Physikalische Fensterstruktur, inhaltlich aber mit der genannten abweichenden Interpretation, definieren wir

```

Type Physik_Informationseinheiten = Record
    Gerät: Physik_Gerät;
    Nachricht: Text;
    Zuordnung: Arbeitszusammenhang;
    Realisierung: Physik_Realisierung
End

```

Mit diesen Vorbereitungen legen wir dann die physikalische Struktur des Mensch-Rechner-Dialogs fest als

```

Type Physik_Struktur = Record
    Akt_Arbeitszusammenhang: Arbeitszusammenhang;
    Akt_Funktion: Set of Physik_Funktionen;
    Akt_Abstr_Struktur: Abstrakte_Dialogstruktur;
    Physik_Fenster: Set of Physik_Fensterstruktur;
    Physik_Einheiten: Set of Physik_Informationseinheiten
End

```

und setzen für die Eingaben des Benutzers:

```

Type Benutzereingabe = Record
    Funktion: Set of Physik_Funktionen;
    Nachricht: Text
End

```

Die physikalische Struktur wird innerhalb des Systems von der **Geräteebene** verwaltet, die für die Transformationen zur Herstellung von Geräteunabhängigkeit zuständig ist. Innerhalb der die Geräteebene bildenden Instanzen AKE, AKA, AKGE und AKGA (vgl. Abb. 13.6) wird die physikalische Struktur in eine abstrakte, geräte- und anwendungsunabhängige Form umgewandelt. Diese

abstrakte Form unterteilen wir in eine Layout-Ebene und in eine logische Ebene. Zur **Layout-Ebene** gehören die Instanzen AKWE, AKDA und AKWA, die Nachrichten in geräteunabhängiger Form bearbeiten, aber durch Trennung von Fensterstruktur und Informationseinheiten noch eine (abstrakte) Aufteilung der zu übergebenden Nachrichten hinsichtlich Bildschirmorientierung oder andere E/A-Geräte vornehmen.

Ausgangspunkt für die Layout-Ebene ist die sogenannte **Layout-Struktur**, die wie die physikalische Struktur aufgebaut ist, aber mit abstrakten Funktionen (die GKS - Ein- und Ausgabefunktionen; vgl Kapitel 10), abstrakten Fenstern und abstrakten Informationseinheiten anstelle der physikalischen.

Type Abstr_Funktionen = (...)

dient der Aufzählung der abstrakten E/A-Funktionen von GKS,

Type Logische_Geräte = (Bildschirm, Drucker, Sprache, Ton, Modem, ..., Tastatur, Maus, Sprache, Funktionstastatur, Digitalisierer, ...)

ist die (unvollständige) Aufzählung der für einen Dialog möglichen E/A-Geräte,

Type Realisierungsform = **Record**
 Schriftart : (...);
 Groß_und_Klein : (...);
 Farbe : (...);
 Helligkeit : (...);
 Blinken : (...);
 Bewegung : (...);

End

für Ausprägungen von Realisierungsaspekten auf dieser Layout-Ebene (Der Typ stimmt bei dieser unvollständigen Definition mit dem Typ Physik_Realisierung überein, bei weiterer Detaillierung der Definition ergeben sich ebenenspezifische Unterschiede, z.B im Wertevorrat.).

Type Abstr_Fensterstruktur = **Record**
 Gerät: Logische_Geräte;
 Nachricht: Text;
 Zuordnung: Arbeitszusammenhang;
 Realisierung: Realisierungsform
End;

Abstr_Informationseinheiten = **Record**
 Gerät: Logische_Geräte;
 Nachricht: Text;
 Zuordnung: Arbeitszusammenhang;
 Realisierung: Realisierungsform
End

sind die Gegenstücke der entsprechenden physikalischen Strukturen auf der Layout-Ebene. Damit ist dann

Type Layout_Struktur = **Record**

```
Akt_Arbeitszusammenhang: Arbeitszusammenhang;
Akt_Funktion: Set of Abstrakte_Funktionen;
Akt_Abstr_Struktur: Abstrakte_Dialogstruktur;
Abstr_Fenster: Set of Abstr_Fensterstruktur;
Abstr_Einheiten: Set of Abstr_Informationseinheiten
End
```

Bestandteil der Datenbasis der Layout-Ebene und gleichzeitig auch Übergabeobjekt zur physikalischen Ebene. Weiter definieren wir zur Übergabe von der physikalischen zur Layout-Ebene

Type Layout_Eingabe = **Record**

```
Funktion: Set of Abstrakte_Funktionen;
Nachricht: Text
End
```

Während bei der physikalischen Struktur die Realisierung auf tatsächlich vorhandene Geräte bezogen wird, ist auf der Layout-Ebene die Realisierung losgelöst von Geräten allgemein z.B. Fenstergröße, Fensterhierarchie, Fensterart (Pop-Up, Pull-Down, Permanent), Schriftarten, Farben, Größen usw. zu verstehen.

Noch eine Stufe abstrakter ist die Realisierungskomponente auf der sogenannten **logischen Ebene**. Hier wird entsprechend des Benutzers und des Dialogtyps (sein Arbeitszusammenhang) die Gestalt der Benutzerschnittstelle abstrakt festgelegt. Dazu gehören die Programmkonzeption (soweit nicht Bestandteil eines Anwendungsprogramms), die abstrakte Dialogstruktur, die Konkretisierung und allgemeine Konzepte für die Realisierung (Hervorhebungen, Fensteranordnungen, Verteilung auf Ausgabegeräte usw.). Wir legen dazu dieser Ebene folgende Datenstrukturen zugrunde:

Type Logische_Funktionen = (...);

{als Aufzählung aller im System möglichen Kontroll-, Gestaltungs-, Meta- und Anwendungsfunktionen}

Konkretisierungsmöglichkeiten = **Record**

```
Vokabular : ( ... );
Inneres_Format : ( ... );
Äußeres_Format : ( ... );
Formale_Redundanz : ( ... );
Ausgabeformat : ( ... );
Semantik : ( ... );
Interpretationsvorschrift : ( ... );
Ausdrucksform : ( ... )
End;
```

```

Realisierungsmöglichkeiten = Record
    Konzept1: ( ... );
    ...
    KonzeptN: ( ... )
End;
{als Grundlage möglicher Konkretisierungen und Realisierungen}

```

```

Logische_Informationseinheiten = Record
    Gerät: Logische_Geräte;
    Nachricht: Text;
    Zuordnung: Arbeitszusammenhang;
    Konkretisierung: Konkretisierungsmöglichkeiten;
    Realisierung: Realisierungsmöglichkeiten
End;
{zur Strukturierung von Information hinsichtlich ihrer Zugehörigkeit zu einem
Arbeitszusammenhang, für Angaben zu ihrer Konkretisierung und Realisierung}

```

```

Logische Struktur = Record
    Akt_Arbeitszusammenhang: Arbeitszusammenhang;
    Akt_Funktion: Set of Logische_Funktionen;
    Akt_Abstr_Struktur: Abstrakte_Dialogstruktur;
    Log_Einheiten: Set of Logische_Informationseinheiten
End

```

Zur logischen Ebene gehören die Instanzen AK1U, AK1N, AK2, AK3N, AK4N, SK1N, SK1U, SK2, GK1U, GK1N, GK2 aus Abb. 13.6. Ihre Aufgabe besteht außer der Festlegung der Benutzerschnittstelle auch in der Transformation der Nachrichten in eine anwendungsspezifische Form. Neben "Wissen" über den Benutzer muß diese Ebene auch "Wissen" über aktuelle Anwendungen besitzen.

Die logische Struktur wird für Systemausgaben zur weiteren Bearbeitung an die Layout-Ebene übergeben. Von der Layout_Ebene aus erreicht die ursprüngliche Benutzereingabe die logische Ebene als logische Eingabe in der Form

```

Type Logische_Eingabe = Record
    Funktion: Set of Logische_Funktionen
    Nachricht: Text;
    Akt_Dialogform: Dialogformen;
    Akt_Arbeitszusammenhang: Arbeitszusammenhang
End

```

Diese Angaben werden analysiert und - falls einer der nachgelagerten Rechner angesprochen wurde - in eine anwendungsabhängige Form überführt und an die sogenannte **Bearbeitungsebene** weitergeleitet. Auf dieser Ebene findet die eigentliche Bearbeitung einer Benutzernachricht statt. Wie dies geschieht ist problembezogen und wird nicht weiter betrachtet. Notwendig ist lediglich die vom Benutzer ausgegangene Nachricht, die zu bearbeitenden Funktionen, sowie der gültige Arbeitszusammenhang. Auf dieser Grundlage werden neue Nachrichten generiert. Wir unterscheiden bei der Bearbeitungsebene somit:

Type Bearbeitungsfunktionen = (...);
 {als Menge der für diese Ebene gültigen Funktionen}

Bearbeitungsstruktur = **Record**
 Nachricht: Text;
 Funktion: **Set of** Bearbeitungsfunktionen;
 Akt_Arbeitszusammenhang: Arbeitszusammenhang
End

14.2. Algorithmen und Typen zur Beschreibung der Dialogeröffnung

Wir beginnen die algorithmische Beschreibung des Systemmodells aus Abb. 13.6 mit der Definition von Typen und Algorithmen für Kanäle und Instanzen, die während der Phase der Dialogeröffnung vom Nachrichtenfluß betroffen sind. Zunächst sind zu nennen:

Type Zustandsmarke_mez = (Dialog_noch_nicht_eröffnet);
 Zustandsmarke_mz = (Dialog_eröffnet, Dialog_beendet);
 Übergabemarke_mae = **Record**
 Funktion : **Set of** Physik_Funktionen
 Nachricht : Text
End

sowie zum Start des Dialogs durch den Benutzer die

Procedure Verarbeitungsvorschrift_ME;
Begin
 Zustandsmarke_mz := Dialog_eröffnet;
 Übergabemarke_mae.Funktion := [Dialogeröffnung_Arbeitsplatzrechner];
 Übergabemarke_mae.Nachricht := 'Dialogprogramm mit Anwendungen auf dem Arbeitsplatzrechner ist gestartet'
End

in der bereits in Kapitel 11 beschriebenen Form. Das Schalten der rechnerseitigen Eröffnungsinstanz ME führt einerseits zu einer Meldung an den Benutzer (/Saathoff82/: "Eröffnungsquittung") und andererseits zu Markierungen der Zustandskanäle des Arbeitsplatzrechners, deren Struktur deshalb zunächst durch Typenvereinbarungen festgelegt werden muß. Wir beginnen mit

Type Log_Funktionsumfang = **Set of** Logische Funktionen;
 Bearb_Funktionsumfang = **Set of** Bearbeitungsfunktionen;
 {zur Zusammenfassung von Funktionen}
 Adressat = (Benutzer, Arbeitsplatzrechner, Schulrechner, Großrechner,
 Nachbar1, ..., NachbarN);
 Hierarchie = Arbeitsplatzrechner .. Großrechner;
 {um die verschiedenen Ansprechpartner bei Dialogschritten unterscheiden zu können}

```

Benutzermodell = ( ... );
{ zur Speicherung von Wissen über den Benutzer}
Bearbeitungsmodell = Record
    Inhalt : Bearbeitungsstruktur;
    Zustand : 1.. 4;
    Partner : Adressat
End;
Bearbeitungsstand = Array[Anwendungszusammenhang] of Bearbeitungsmodell

```

und setzen dann in Anlehnung an das im letzten Abschnitt Gesagte:

```

Type Zustandsmarke_akz1 = Record
    Dialogpartner : Adressat;
    Dialogrechner: Hierarchie;
    Benutzerwissen: Benutzermodell;
    Dialogwissen: Array[Arbeitszusammenhang] of
        Log_Funktionsumfang;
    Log_Strukt: Logische_Struktur;
    Abstrakt: Array[1..Maxint] of Abstrakte Dialogstruktur;
    Konkret: Array[1..Maxint] of Konkretisierungsmöglichkeiten;
    Rea: Array[1..Maxint] of Realisierungsmöglichkeiten
End;
Zustandsmarke_akz2 = Record
    Dialogpartner: Adressat;
    Dialogrechner: Hierarchie;
    Zwischenspeicher: Record
        log_Funk: Set of Logische_Funktionen;
        Nachricht: Text
    End;
    Anwendungswissen: Array[Arbeitszusammenhang] of
        Bearb_Funktionsumfang;
    Akt_Arbeitszusammenhang: Arbeitszusammenhang
End;
Zustandsmarke_akzw = Record
    Layout_Strukt: Layout_Struktur;
    Rea: Array[1..Maxint] of Realisierungsformen
End;
Zustandsmarke_akzg = Record
    Physik_Strukt: Physik_Struktur;
    Rea: Array[1..Maxint] of Physik_Realisierungen
End;
Zustandsmarke_az = Record
    Dialogpartner : Adressat;
    Dialogrechner: Hierarchie;
    Akt_Anwendung: Arbeitszusammenhang;
    Anwendungsumgebung : Set of Arbeitszusammenhang;
    Anwendungsinhalt: Bearbeitungsstand
End

```

Weiter ist die Struktur der Übergabemarke akü24 festzulegen. Ihre Aufgabe innerhalb der logischen Ebene ist das zur Verfügung stellen von Nachrichten

und logischen Funktionen für die weitere Verarbeitung, wobei ihr insbesondere eine Verteilfunktion von vier vorgelagerten zu sechs nachgelagerten Instanzen zukommt. Wir setzen:

```
Type übergabemarke_akü24 = Record
    Funktion : Set of Logische_Funktionen;
    Nachricht : Text;
    Dialogpartner: Adressat;
    Akt_Arbeitszusammenhang: Arbeitszusammenhang;
    Akt_Arbeitsumgebung: Set of Arbeitszusammenhang
End;
```

Auf der Grundlage dieser Typenvereinbarungen folgt nun die Systemeröffnung durch die Instanz AE. Im Modell für das einfache CUU-Programm des Abschnitts 11.4 konnten wir durch Strukturierung des Programmablaufs Funktionen und Zustände angeben und dann das Dialogverhalten explizit durch Instanzenvorschriften und Attributsmarken beschreiben. Eine analoge Vorgehensweise ist hier nicht möglich. Vielmehr muß durch die Algorithmen allgemein jeder konkrete Dialog innerhalb des Systems modelliert werden können. Aus diesem Grund ist eine sehr allgemeine Beschreibung des prinzipiellen Ablaufs innerhalb der Instanzen notwendig; alle realisierungsnahen und inhaltlichen Aspekte werden von uns deshalb durch Prozeduren und Funktionen nachgebildet, deren Gestalt wir nicht weiter betrachten.

```
Procedure Verarbeitungsvorschrift_AE;
    var Zähler: Integer;
        Typ: Arbeitszusammenhang;
        Arbeitsumgebung: Set of Arbeitszusammenhang;
Function Eröffnungsnachricht : Text;
    {Das Unterprogramm liefert die an den Benutzer zu leitende Nachricht}
Procedure Init_Benutzerwissen(Var Benutzer : Benutzermodell);
    {Durch die Prozedur wird für den Start des Systems ein anfängliches Benutzermodell vorgegeben. Dabei ist es denkbar, daß hierfür auf ein bestehendes, durch die Systemeröffnung wieder verfügbares Modell zurückgegriffen wird, falls der Benutzer bereits früher einmal mit dem System gearbeitet hat. Daß in der Zwischenzeit Teile der Handhabungsmöglichkeiten vom Benutzer wieder vergessen wurden, kann z.B. durch die seit dem letzten Arbeiten verstrichene Zeit berücksichtigt werden}
Procedure Init_Log_Funktionsumfang (Dialogtyp: Arbeitszusammenhang;
    var System: Log_Funktionsumfang);
    {In Abhängigkeit des Dialogtyps werden die zugehörigen logischen Funktionen zu einer Menge zusammengefaßt}
Procedure Init_Abstrakte_Dialogstruktur (Var Abstr_Dialogstruktur:
    Array[1..Maxint] of Abstrakte_Dialogstruktur);
    {Alle zum Zeitpunkt der Dialogeröffnung sinnvollen abstrakten Dialogstrukturen werden zur Verfügung gestellt}
Procedure Init_Konkretisierungsmöglichkeiten (Var Konkretisierung:
    Array[1..Maxint] of Konkretisierungsmöglichkeiten);
    {Alle zum Zeitpunkt der Dialogeröffnung sinnvollen Konkretisierungsmöglichkeiten werden zur Verfügung gestellt}
Procedure Init_Realisierungsmöglichkeiten (Var Realisierung:
```

```

                                Array[1..Maxint] of Realisierungsmöglichkeiten);
    {Alle zum Zeitpunkt der Dialogeröffnung sinnvollen Realisierungsmöglichkeiten werden
    zur Verfügung gestellt}
Procedure Init_Bearb_Funktionsumfang (Dialogtyp: Arbeitszusammenhang;
                                Var System: Bearb_Funktionsumfang);
    {Abhängig vom Dialogtyp werden die zugehörigen Bearbeitungsfunktionen zu einer
    Menge zusammengefaßt}
Procedure Init_Realisierungsform (Var Realisierung: Array[1..Maxint] of Realisierungsform);
    {Alle zum Zeitpunkt der Dialogeröffnung sinnvollen Realisierungsformen werden zur
    Verfügung gestellt}
Procedure Init_Physik_Realisierungen (Var Realisierung:
                                Array[1..Maxint] of Physik_Realisierung);
    {Alle zum Zeitpunkt der Dialogeröffnung sinnvollen physikalischen Realisierungen
    werden zur Verfügung gestellt}
Procedure Init_Arbeitszusammenhang ( Arb_Zus: Arbeitszusammenhang;
                                Var Bearb_Strukt: Bearbeitungsstruktur);
    {Für einen Dialogtyp wird der letzte Bearbeitungsstand ermittelt}
Procedure Init_Akt_Arbeitszusammenhang (Var Typ: Arbeitszusammenhang);
    {Das durch die Eröffnung festgelegte aktuelle Anwendungsprogramm wird bestimmt}
Procedure Init_Umgebung (Var Dialogtypen: Set of Arbeitszusammenhang);
    { Die durch die Eröffnung festgelegte aktuelle Arbeitsumgebung mit den dem System
    bekannten Dialogtypen wird bestimmt}

Begin {Verarbeitungsvorschrift_AE}
Init_Umgebung(Arbeitsumgebung);
With Zustandsmarke_akz1 do
    Begin Dialogrechner:= Arbeitsplatzrechner;
        Init_Benutzerwissen (Benutzerwissen);
        For Typ:=Kontrolle to AnwendungN do
            If Typ In Arbeitsumgebung
                then Init_Log_Funktionsumfang (Typ,Dialogwissen[Typ]);
        Init_Abstrakte_Dialogstruktur(Abstrakt);
        Log_Strukt.Log_Einheiten:=[];
        Log_Strukt.Akt_Abstr_Struktur:=[];
        Init_Konkretisierungsmöglichkeiten(Konkret);
        Init_Realisierungsmöglichkeiten (Rea)
    End;
With Zustandsmarke_akz2 do
    Begin Dialogpartner:=Benutzer;
        Dialogrechner:= Arbeitsplatzrechner;
        For Typ:=Kontrolle to AnwendungN do
            If Typ In Arbeitsumgebung
                then Init_Bearb_Funktionsumfang (Typ,Anwendungswissen[Typ]);
        Init_Akt_Arbeitszusammenhang (Akt_Arbeitszusammenhang)
    End;
With Zustandsmarke_akzw do
    Begin Layout_Strukt_Abstr_Fenster:=[];
        Layout_Strukt_Abstr_Einheiten:=[];
        Init_Realisierungsform (Rea)
    End;
With Zustandsmarke_akzw do
    Begin Physik_Strukt.Physik_Fenster:=[];

```

```

    Physik_Strukt.Physik_Einheiten:=[];
    Init_Physik_Realisierungen (Rea)

End;
Init_Anwendung(Typ);
With Zustandsmarke_az do
  Begin Dialogpartner := Benutzer;
    Dialogrechner:= Arbeitsplatzrechner;
    For Typ:= Kontrolle To AnwendungN Do
      If Typ In Arbeitsumgebung
        Then Begin
          Init_Arbeitszusammenhang(Typ, Bearb_Struktur);
          Anwendungsstand[Typ].Inhalt := Bearb_Struktur;
          Anwendungsstand[Typ].Zustand:= 4;
          Anwendungsstand[Typ].Partner:= Benutzer
        End;
        { Für alle in der Arbeitsumgebung vorkommenden Dialogtypen
          wird der letzte Bearbeitungsstand ermittelt und gespeichert}
      Init_Akt_Arbeitszusammenhang(Akt_Anwendung);
      Anwendungsumgebung:=Arbeitsumgebung
    End;
With Übergabemarke_akü24 do
  Begin Funktion := [Eröffnungsquittung];
    Nachricht := Eröffnungsnachricht;
    Dialogpartner := Benutzer;
    Init_Akt_Arbeitszusammenhang(Akt_Arbeitszusammenhang);
    Akt_Arbeitsumgebung := Arbeitsumgebung
  End
End; {Verarbeitungsvorschrift_AE}

```

Nach dem Schalten von AE wird die Marke auf akü24 von AK3N aufgenommen und mit Hilfe der Informationen auf akz1 die logische Struktur erzeugt. Dies geschieht in der Form, daß entsprechend den zu diesem Zeitpunkt gültigen Arbeitsumgebungen logische Informationseinheiten in Abhängigkeit vom Benutzerwissen und den Konkretisierungs- und Realisierungsmöglichkeiten aufgebaut werden. In der Regel wird dies eine Reihe von Kontroll-, Gestaltungs- und Metafunktionen sein, dazu eine (die gestartete) Anwendung, sowie einige Standardanwendungen (Taschenrechner, Datum, Uhrzeit). Um diesen allgemeinen Vorgang algorithmisch beschreiben zu können, setzen wir wieder eine Reihe nicht näher bestimmter Prozeduren voraus.

Da die logische Struktur sowohl Bestandteil der Zustandsmarke akz1 ist, als auch für die weitere Verarbeitung von AKWA benötigt wird, definieren wir zunächst die Struktur der Übergabemarke akü23 als:

```

Type Übergabemarke_akü23 = Record
  Log_Strukt: Logische_Struktur;
  Akt_Arbeitsumgebung: Set of Arbeitszusammenhang
End

```

und dann als Verarbeitungsvorschrift für die Instanz AK3N:

```

Procedure Verarbeitungsvorschrift_AK3N;
  var Dialogtyp: Arbeitszusammenhang;
      Alt_Log_Inf_Ein, Neu_Log_Inf_Ein: Set of Logische_Informationseinheiten;
      Alt_Log_Funk, Neu_Log_Funk: Set of Logische_Funktionen;
Function Dialogtyp_von_Akt_Funktion_betroffen: Boolean;
  {Feststellung, ob eine Funktion einen Dialogtyp betrifft}
Procedure Bestimme_Akt_Abstr_Dialogstruktur (Benutzer: Benutzermodell;
      Abstr_Dialogstruk: Array[1..Maxint] of Abstrakte_Dialogstruktur;
      Var Akt_Abstr_Dialogstruktur: Abstrakte_Dialogstruktur);
  {In Abhängigkeit vom Benutzer und den generellen Möglichkeiten zur Bildung
  einer abstrakten Dialogstruktur wird die zu diesem Zeitpunkt optimale aktuelle
  abstrakte Dialogstruktur festgelegt}
Procedure Bestimme_Log_Informationseinheiten
  (Benutzer: Benutzermodell;
   Akt_Funk: Set of Logische_Funktionen;
   Akt_Abstr_Dialogstruktur: Abstrakte_Dialogstruktur;
   Typ: Arbeitszusammenhang;
   System: Log_Funktionsumfang;
   Inhalt: Text;
   Konkr_Mögl: Array[1..Maxint] of Konkretisierungsmöglichkeiten;
   Realis_Mögl: Array[1..Maxint] of Realisierungsmöglichkeiten;
   Var Log_Inf_Ein_Alt, Log_Inf_Ein_Neu:
       Set of Logische_Informationseinheiten;
   Var Log_Funk_Alt, Log_Funk_Neu: Set of Logische_Funktionen);
  {Zu einem in diesem Zeitpunkt vorgesehenen Arbeitszusammenhang bestimmt
  die Prozedur eine Menge logischer Informationseinheiten, abhängig vom
  Benutzer, von der aktuellen abstrakten Dialogstruktur, von den innerhalb des
  Arbeitszusammenhangs definierten Funktionen und den generell möglichen
  Konkretisierungen und Realisierungen. Die dem Benutzer zu übergebende
  Nachricht wird auf die logischen Informationseinheiten aufgeteilt. Logische
  Informationseinheiten induzieren in der Regel logische Funktionen, z.B. daß etwas
  auf dem Bildschirm in bestimmter Art und Weise dargestellt wird. Diese Funktionen
  werden von der Prozedur bestimmt und übergeben}

Begin {Verarbeitungsvorschrift_AK3N}
  Zustandsmarke_akz1.Dialogpartner:= Übergabemarke_akü24.Dialogpartner;
With Zustandsmarke_akz1.Log_Strukt do
  Begin
    Akt_Arbeitszusammenhang:= Übergabemarke_akü24_Akt_Arbeitszusammenhang;
    Akt_Funktion:= Übergabemarke_akü24.Funktion;
    If (Akt_Abstr_Dialogstruktur = []) or (Verändere_Akt_Abstr_Struktur in Akt_Funktion)
      Then Bestimme_Akt_Abstr_Dialogstruktur ( Zustandsmarke_akz1.Benutzerwissen,
      Zustandsmarke_akz1.Abstrakt,
      Akt_Abstr_Dialogstruktur);
      {Die abstrakte Dialogstruktur wird nur bei Dialogeröffnung oder im Falle
      einer notwendigen Anpassung an verändertes Benutzerwissen bestimmt}
  For Dialogtyp:= Kontrolle to AnwendungN do
    If (Dialogtyp in Übergabemarke_akü24.Akt_Arbeitsumgebung)
      and (Dialogtyp_von_Akt_Funktion_betroffen)
      Then Begin
        Bestimme_Log_Informationseinheiten
          (Zustandsmarke_akz1.Benutzerwissen,

```

```

    Akt_Funktion,
    Akt_Abstr_Dialogstruktur,
    Dialogtyp,
    Zustandsmarke_akz1.Dialogwissen[Dialogtyp],
    Übergabemarke_akü24.Nachricht;
    Zustandsmarke_akz1.Konkret,
    Zustandsmarke_akz1.Rea,
    Alt_Log_Inf_Ein,
    Neu_Log_Inf_Ein,
    Alt_Log_Funk,
    Neu_Log_Funk);
    Logische_Einheiten:=Logische_Einheiten - Alt_Log_Inf_Ein;
    Logische_Einheiten:=Logische_Einheiten + Neu_Log_Inf_Ein;
    Akt_Funktion:=Akt_Funktion - Alt_Log_Funk;
    Akt_Funktion:=Akt_Funktion + Neu_Log_Funk
    {Wenn nicht ein völliger Neuaufbau der logischen Struktur erfolgt, gibt es
    bereits logische Einheiten und Funktionen in akz1. Diese können von
    Veränderungen betroffen sein. Wir drücken durch die Anweisungen aus,
    daß zunächst die alten ungültigen Elemente der Mengen gelöscht, bevor
    die veränderten neuen eingefügt werden.}
  End
  {Im Falle der Dialogeröffnung ist Logische_Einheiten gleich [] und alle
  Dialogtypen sind von Akt_Funktion betroffen. Die logischen
  Informationseinheiten werden also vollständig aufgebaut. In allen
  anderen Fällen gibt es bereits eine logische Struktur auf akz1, die lediglich dem
  neuen Zustand angepaßt wird}

End;
Übergabemarke_akü23.Log_Strukt:= Zustandsmarke_akz1.Log_Strukt;
Übergabemarke_akü23.Akt_Arbeitsumgebung:=
    Übergabemarke_akü24.Akt_Arbeitsumgebung
End {Verarbeitungsvorschrift_AK3N}

```

Die Marke auf akü23 mit der logischen Struktur bildet die Grundlage für die Arbeit von der Instanz AKWA, die die logische Struktur in eine Layout_Struktur transformiert. Dazu sind zu den logischen Funktionen abstrakte Funktionen zu bestimmen, die logischen Informationseinheiten zu splitten und dabei die Konkretisierungs- und Realisierungsmöglichkeiten in - realisierungsnähere - Realisierungsformen zu überführen. Das Ergebnis ist sowohl bei akzw zu speichern wie an akü22 zu übergeben. Wir definieren deshalb:

```

Type Übergabemarke_akü22 = Record
    Layout_Strukt: Layout_Struktur;
    Akt_Arbeitsumgebung: Set of Arbeitsumgebung
End

```

Für die Verarbeitungsvorschrift der Instanz AKWA gehen wir wie gewohnt mit nicht näher erklärten Prozeduren vor, um die inhaltlichen Aspekte der Transformationen algorithmisch darstellen zu können.

Procedure Verarbeitungsvorschrift_AKWA;

var Dialogtyp: Arbeitszusammenhang;
 Log_Inf_Einh, Log_B_Inf_Einh: **Set of** Logische Informationseinheiten;
 Alt_Ab_Fenster, Neu_Ab_Fenster: **Set of** Abstr_Fensterstruktur;
 Alt_Ab_Inf_Einh, Neu_Ab_Inf_Einh: **Set of** Abstr_Informationseinheiten;
 Alt_Ab_Funk, Neu_Ab_Funk: **Set of** Abstrakte_Funktionen;

Function Dialogtyp_von_Akt_Funktion_betroffen: Boolean;
 {Feststellung, ob eine Funktion einen Dialogtyp betrifft}

Procedure Bestimme_Abstr_Funktionen (Log_Funk: **Set of** Logische_Funktionen,
Var Abstr_Funk: **Set of** Abstr_Funktionen);
 {Zu einer Menge logischer Funktionen wird die sie darstellende Menge abstrakter
 Funktionen gebildet}

Procedure Bestimme_Abstr_Fenster
 (Log_Inf_Einh: **Set of** Logische_Informationseinheiten;
 Realis_Form: **Array**[1..Maxint] **of** Realisierungsformen;
Var Ab_Fen_Alt, Ab_Fen_Neu: **Set of** Abstrakte_Fensterstruktur;
Var Ab_Funk_Alt, Ab_Funk_Neu: **Set of** Abstr_Funktionen);
 {Für diejenigen logischen Informationseinheiten, die sich auf das logische Gerät
 "Bildschirm" beziehen, wird - innerhalb eines Arbeitszusammenhangs - eine Menge
 von abstrakten Fenstern gebildet, deren Realisierungsform von den logischen
 Informationseinheiten und den generell möglichen Formen abhängt. Die aus den
 abstrakten Fenstern resultierenden abstrakten Funktionen werden übergeben}

Procedure Bestimme_Abstr_Einheiten
 (Log_Inf_Einh: **Set of** Logische_Informationseinheiten;
 Realis_Form: **Array**[1..Maxint] **of** Realisierungsformen;
Var Ab_Einh_Alt, Ab_Einh_Neu: **Set of**
 Abstrakte_Informationseinheiten;
Var Ab_Funk_Alt, Ab_Funk_Neu: **Set of** Abstr_Funktionen);
 {Für diejenigen logischen Informationseinheiten, die sich nicht auf das logische Gerät
 "Bildschirm" beziehen, wird - innerhalb eines Arbeitszusammenhangs - eine Menge
 von abstrakten Einheiten gebildet, deren Realisierungsform von den logischen
 Informationseinheiten und den generell möglichen Formen abhängt. Die aus den
 abstrakten Einheiten resultierenden abstrakten Funktionen werden übergeben}

Procedure Ermittle_Log_Einheiten
 (Typ: Arbeitszusammenhang;
 Log_Funk: **Set of** Logische_Funktionen;
 Log_Inf_Einh: **Set of** Logische_Informationseinheiten;
Var Akt_Log_Einh: **Set of** Logische_Informationseinheiten);
 {Die Prozedur ermittelt innerhalb eines Arbeitszusammenhangs die Menge der
 logischen Einheiten, die von den aktuellen Funktionen betroffen sind}

Procedure Ermittle_Bildschirm_Einheiten
 (Log_Inf_Einh: **Set of** Logische_Informationseinheiten;
Var Log_B_Inf_Einh: **Set of** Logische_Informationseinheiten);
 {Die Prozedur ermittelt innerhalb eines Arbeitszusammenhangs die Menge der
 logischen Einheiten, die für das logische Gerät "Bildschirm" vorgesehen sind}

Begin {Verarbeitungsvorschrift_AKWA}

With Zustandsmarke_akzw.Layout_Strukt **do**

Begin

Akt_Arbeitszusammenhang:=

Übergabemarke_akü23.Log_Strukt.Akt_Arbeitszusammenhang;

Bestimme_Abstr_Funktionen (Übergabemarke_akü23.Log_Strukt.Akt_Funktion,

```

Akt_Funktion);
For Dialogtyp:= Kontrolle to AnwendungN do
  If (Dialogtyp In Übergabemarke_akü23.Akt_Arbeitsumgebung)
    and (Dialogtyp_von_Akt_Funktion_betroffen)
  Then
    Begin
      Ermittle_Log_Einheiten
        (Dialogtyp,
         Übergabemarke_akü23.Log_Strukt.Akt_Funktion,
         Übergabemarke_akü23.Log_Strukt.Log_Einheiten,
         Log_Inf_Ein);
      Ermittle_Bildschirm_Einheiten (Log_Inf_Ein,Log_B_Inf_Ein);
      Bestimme_Abstr_Fenster (Log_B_Inf_Ein,
        Zustandsmarke_akzw.Rea,
        Alt_Ab_Fenster,
        Neu_Ab_Fenster,
        Alt_Ab_Funk,
        Neu_Ab_Funk);
      Abstr_Fenster:=Abstr_Fenster - Alt_Ab_Fenster;
      Abstr_Fenster:=Abstr_Fenster + Neu_Ab_Fenster;
      Akt_Funktion:=Akt_Funktion - Alt_Ab_Funk;
      Akt_Funktion:=Akt_Funktion + Neu_Ab_Funk;
      Log_Inf_Ein:= Log_Inf_Ein - Log_B_Inf_Ein;
      Bestimme_Abstr_Einheiten (Log_Inf_Ein,
        Zustandsmarke_akzw Rea,
        Alt_Ab_Inf_Ein,
        Neu_Ab_Inf_Ein,
        Alt_Ab_Funk,
        Neu_Ab_Funk);
      Abstr_Einheiten:=Abstr_Einheiten - Alt_Ab_Inf_Ein;
      Abstr_Einheiten:=Abstr_Einheiten + Neu_Ab_Inf_Ein;
      Akt_Funktion:=Akt_Funktion - Alt_Ab_Funk;
      Akt_Funktion:=Akt_Funktion + Neu_Ab_Funk
    End;
  { Auch hier wurde das Ersetzen von alten Fenstern, bzw. Informations-
    einheiten durch neue und das Ersetzen von Funktionen, die sich auf den
    alten Zustand beziehen, durch jeweils zwei Anweisungen ausgedrückt, vgl.
    Verarbeitungsvorschrift AK3N }
  Übergabemarke_akü22.Layout_Strukt:= Zustandsmarke_akzw.Layout_Strukt;
  Übergabemarke_akü22.Akt_Arbeitsumgebung:=
    Übergabemarke_akü23.Akt_Arbeitsumgebung
End {Verarbeitungsvorschrift_AKWA}

```

Nach Umwandlung der logischen Struktur in geräteunabhängige Layout-Struktur zur Manipulation von Fenstern erfolgt nun eine Transformation in geräteabhängige Form durch die Instanz AKGA. Diese nimmt mit der Layout-Struktur die abstrakten Funktionen auf, ermittelt auf der Grundlage der Vorgaben des Übergabekanals und der generellen Realisierungsvorgaben ihres Zustandskanals physikalische Fenster, physikalische Informationseinheiten und physikalische Ausgabefunktionen. Wir beschreiben die Wirkung der Instanz algorithmisch durch folgende Prozedur, wobei zuvor

```

Type Übergabemarke_akü21 = Record
    Physik_Strukt: Physik_Struktur;
    Akt_Arbeitsumgebung: Set of Arbeitsumgebung
End

```

gesetzt wird.

```

Procedure Verarbeitungsvorschrift_AKGA;
    var Dialogtyp: Arbeitszusammenhang;
        Ab_Fen: Set of Abstr_Fensterstruktur;
        Ab_Inf_Ein: Set of Abstr_Informationseinheiten;
        Alt_Phys_Fen, Neu_Phys_Fen: Set of Physik_Fensterstruktur;
        Alt_Phys_Inf_Ein, Neu_Phys_Inf_Ein: Set of Physik_Informationseinheiten;
        Alt_Phys_Funk, Neu_Phys_Funk: Set of Physik_Funktionen;
    Function Dialogtyp_von_Akt_Funktion_betroffen: Boolean;
        {Feststellung, ob eine Funktion einen Dialogtyp betrifft}
    Procedure Bestimme_Physik_Funktionen (Abstr_Funk: Set of Abstr_Funktionen;
        Var Phys_Funk: Set of Physik_Funktionen);
        {Zu einer Menge abstrakter Funktionen wird die sie darstellende Menge
        physikalischer Funktionen gebildet}

```

```

Procedure Bestimme_Physik_Fenster
    (Abstr_Fen: Set of Abstr_Fensterstruktur;
    Phys_Realis: Array[1..Maxint] of Physik_Realisierungen;
    Var Phys_Fen_Alt, Phys_Fen_Neu: Set of Physik_Fensterstruktur;
    Var Phys_Funk_Alt, Phys_Funk_Neu: Set of Physik_Funktionen);
    {Zu einer Menge abstrakter Fenster wird in Abhängigkeit von den Möglichkeiten
    zur physikalischen Realisierung eine Menge physikalischer Fenster ermittelt und
    diese, sowie die aus ihnen resultierenden physikalischen Funktionen werden
    übergeben}

```

```

Procedure Bestimme_Physik_Einheiten
    (Abstr_Inf_Einh: Set of Abstr_Informationseinheiten;
    Phys_Realis: Array[1..Maxint] of Physik_Realisierungen;
    Var Phys_Inf_Ein_Alt, Phys_Inf_Ein_Neu: Set of
        Physik_Informationseinheiten;
    Var Phys_Funk_Alt, Phys_Funk_Neu: Set of Physik_Funktionen);
    {dito für physikalische Informationseinheiten}

```

```

Procedure Ermittle_Abstr_Fenster (Typ: Arbeitszusammenhang;
    Abstr_Funk: Set of Abstr_Funktionen;
    Abstr_Fen: Set of Abstr_Fensterstruktur;
    Var Akt_Abstr_Fen: Set of Abstr_Fensterstruktur);
    {Die Prozedur ermittelt innerhalb eines Arbeitszusammenhangs die Menge der
    abstrakten Fenster, die von den aktuellen Funktionen betroffen sind}

```

```

Procedure Ermittle_Abstr_Einheiten (Typ: Arbeitszusammenhang;
    Abstr_Funk: Set of Abstr_Funktionen;
    Abstr_Inf_Einh: Set of Abstr_Informationseinheiten
    Var Akt_Abstr_Einh: Set of Abstr_Informationseinheiten);
    {Die Prozedur ermittelt innerhalb eines Arbeitszusammenhangs die Menge der ab-
    strakten Informationseinheiten, die von den aktuellen Funktionen betroffen sind}

```

```

Begin {Verarbeitungsvorschrift_AKGA}
With Zustandsmarke_akzg.Physik_Strukt do

```

```

Begin
Akt_Arbeitszusammenhang:=
    Übergabemarke_akü22.Layout_Struktur.Akt_Arbeitszusammenhang;
Bestimme_Physik_Funktionen (Übergabemarke_akü22.Layout_Strukt.Akt_Funktion,
    - Akt_Funktion);
For Dialogtyp:= Kontrolle to AnwendungN do
    If (Dialogtyp In Übergabemarke_akü22.Akt_Arbeitsumgebung)
        and (Dialogtyp_von_Akt_Funktion_betroffen)
        Then Begin
            Ermittle_Abstr_Fenster (Dialogtyp,
                Übergabemarke_akü22.Layout_Strukt.Akt_Funktion,
                Übergabemarke_akü22.Layout_Strukt.Abstr_Fenster,
                Ab_Fen);
            Bestimme_Physik_Fenster (Ab_Fen,
                Zustandsmarke_akzg.Rea,
                Alt_Phys_Fen,
                Neu_Phys_Fen,
                Alt_Phys_Funk,
                Neu_Phys_Fen);
            Physik_Fenster:=Physik_Fenster - Alt_Phys_Fen;
            Physik_Fenster:=Physik_Fenster + Neu_Phys_Fen;
            Akt_Funktion:=Akt_Funktion - Alt_Phys_Funk;
            Akt_Funktion:=Akt_Funktion + Neu_Phys_Funk;
            Ermittle_Abstr_Einheiten (Dialogtyp,
                Übergabemarke_akü22.Layout_Struktur.Akt_Funktion,
                Übergabemarke_akü22.Layout_Struktur.Abstr_Einheiten,
                Ab_Inf_Ein);
            Bestimme_Physik_Einheiten (Ab_Inf_Ein,
                Zustandsmarke_akzg.Rea,
                Alt_Phys_Inf_Ein,
                Neu_Phys_Inf_Ein,
                Alt_Phys_Funk,
                Neu_Phys_Funk);
            Physik_Einheiten:=Physik_Einheiten - Alt_Phys_Inf_Ein;
            Physik_Einheiten:=Physik_Einheiten + Neu_Phys_Inf_Ein;
            Akt_Funktion:=Akt_Funktion - Alt_Phys_Funk;
            Akt_Funktion:=Akt_Funktion + Neu_Phys_Funk
        End;
Übergabemarke_akü21.Physik_Strukt:= Zustandsmarke_akzg.Physik_Strukt,
Übergabemarke_akü21.Akt_Arbeitsumgebung:=
    Übergabemarke_akü22.Akt_Arbeitsumgebung
End
End {Verarbeitungsvorschrift_AKGA}

```

Bei Beendigung der Prozedur steht die Ausgabenachricht auf dem Kanal akü21 zur Ausgabe zur Verfügung. Die angesprochenen Geräte entnehmen der physikalischen Struktur die sie betreffenden physikalischen Einheiten und übergeben die darin enthaltenen Nachrichten den aufnehmenden Instanzen des Benutzers:

Procedure Verarbeitungsvorschrift_AKA;

Begin

{Ermittle die das Gerät betreffenden physikalischen Einheiten und stelle die Nachrichten auf dem Gerät dar}

End;

Die notwendige Konfliktlösungsvorschrift richtet sich nach den in der physikalischen Struktur angegebenen physikalischen Funktionen (Bildschirmausgabe, Tonausgabe, ...) und verläuft in der Weise, wie in Kapitel 11 für das Beispiel angegeben. Wir wollen sie ebenso wenig weiter betrachten, wie die Verarbeitungsvorschriften der Benutzerinstanzen MK2, MN und MK1 und setzen für das Folgende voraus, daß nach Schalten der Instanzen AKE auf dem Kanal akü11 die dort definierte Struktur

Type Übergabemarke_akü11 = **Record**

Eingabe: Benutzereingabe;

Akt_Arbeitsumgebung: **Set of** Arbeitszusammenhang

End

mit Werten belegt ist. Akt_Arbeitsumgebung kann sich dabei z.B. durch Starten einer neuen Anwendung gegenüber der Systemausgabe verändert haben.

14.3. Algorithmen und Typen zur Beschreibung eines Dialogschritts im normalen Dialog

Die Benutzereingabe vom Kanal akü11 wird von AKGE aufgenommen und mit Hilfe der physikalischen Struktur auf akzg in eine Layout-Eingabe überführt, d.h. die physikalischen Funktionen werden in Layout-Funktionen (also in GKS-Eingabefunktionen) transformiert. Es ist:

Type Übergabemarke_akü12 = **Record**

Lay_Ein: Layout_Eingabe;

Akt_Arbeitsumgebung = **Set of** Arbeitszusammenhang

End

und dann

Procedure Verarbeitungsvorschrift_AKGE;

Procedure Bestimme_Abstr_Funktionen

(Physik_Funk: **Set of** Physik_Funktionen;

Physik_Strukt: Physik_Struktur;

Abstr_Funk: **Set of** Abstr_Funktionen);

{Bestimmung von abstrakten Funktionen aus physikalischen unter Zuhilfenahme der physikalischen Struktur}

Begin {Verarbeitungsvorschrift_AKGE}

Bestimme_Abstr_Funktionen (Übergabemarke_akü11.Eingabe.Funktion,

Zustandsmarke_akzg.Physik_Strukt,

Übergabemarke_akü12.Lay_Ein.Funktion);

Übergabemarke_akü12.Lay_Ein.Nachricht:= Übergabemarke_akü11.Eingabe.Nachricht;

Übergabemarke_akü12.Akt_Arbeitsumgebung:=

Übergabemarke_akü11.Akt_Arbeitsumgebung

End {Verarbeitungsvorschrift_AKGE}

Mit Hilfe der Layout_Eingabe, der aktuellen Arbeitsumgebung und der Layout-Struktur des Zustandskanals akzw muß nun AKWE zwischen einer direkten Ausgabe und einer weiter zu bearbeitenden Eingabe unterscheiden. Im Falle einer direkten Ausgabe müssen von der Instanz die betroffenen abstrakten Fenster und abstrakten Informationseinheiten ermittelt und verändert werden und die derart geänderte Layout-Struktur an AKDA übergeben werden. Im anderen Fall ist die vom Benutzer gewählte Dialogform und der aktuelle Arbeitszusammenhang zu bestimmen, und es sind die abstrakten Funktionen in logische zu überführen, um die Logische Eingabe formulieren zu können. Wir definieren

Type Übergabemarke_akü13 = **Record**

Log_Ein: Logische_Eingabe;

Layout_Strukt: Layout_Struktur;

Akt_Arbeitsumgebung = **Set of** Arbeitszusammenhang

End

und können dann wieder unter Verwendung nicht näher bestimmter Prozeduren für AKWE formulieren:

Procedure Verarbeitungsvorschrift_AKWE;

Var Ab_Funk1, Ab_Funk2: **Set of** Abstr_Funktionen;

Function Direkte_Ausgabe (Abstr_Funk: **Set of** Abstr_Funktionen;

Lay_Struk: Layout_Struktur): **Boolean**;

{Das Programm ermittelt aus den vom Benutzer ausgegangenen Funktionen mit Hilfe der gespeicherten Layout-Struktur, ob es sich bei der Benutzereingabe um eine Aktion handelt, die eine direkte Ausgabe erforderlich macht}

Procedure Verändere_Abstr_Fenster

(Realis_Form: **Array**[1..Maxint] **of** Realisierungsformen;

Abstr_Funk: **Set of** Abstr_Funktionen;

Nachricht: **Text**;

var: Abstr_Fenster: **Set of** Abstr_Fensterstruktur;

var: Akt_Abstr_Funk: **Set of** Abstr_Funktionen);

Procedure Verändere_Abstr_Einheiten

(Realis_Form: **Array**[1..Maxint] **of** Realisierungsformen;

Abstr_Funk: **Set of** Abstr_Funktionen;

Nachricht: **Text**;

var: Abstr_Einheiten: **Set of** Abstr_Informationseinheiten;

var: Akt_Abstr_Funk: **Set of** Abstr_Funktionen);

{Die beiden Prozeduren bestimmen aufgrund der aus der Eingabe resultierenden abstrakten Funktionen diejenigen abstrakten Fenster bzw. Einheiten, die durch die Funktion betroffen sind und verändern innerhalb der Layout-Struktur diese Fenster bzw. Einheiten entsprechend (unter Zuhilfenahme der alten Werte, der Benutzernachricht und der Realisierungsformen). Die durch diese Veränderungen sich ergebenden abstrakten Funktionen (für die Ausgabe) werden ermittelt und übergeben. Der dritte Übergabeparameter dient hier der Ein- und Ausgabe, d.h. eine

Menge Abstrakter Fenster bzw. Einheiten wird vorgegeben und durch die Prozedur verändert.)

```
Procedure Bestimme_Log_Funktionen (Abstr_Funk: Set of Abstr_Funktionen;
                                   Lay_Strukt: Layout_Struktur;
                                   var: Log_Funk: Set of Logische_Funktionen);
{Zu einer Menge abstrakter Funktionen wird die sie darstellende Menge logischer
Funktionen bestimmt. Im Falle fehlerhafter Eingaben wird eine Funktion
"Fehlermeldung" erzeugt.}
```

```
Procedure Bestimme_Akt_Dialogform (Abstr_Funk: Set of Abstr_Funktionen;
                                   Lay_Strukt: Layout_Struktur;
                                   Arb_Umgeb: Set of Arbeitszusammenhang;
                                   var: Akt_Dia_Form: Dialogformen);
{Durch die Prozedur wird die vom Benutzer für seine Eingabe verwendete Dialogform
bestimmt. Dies ist zum einen nötig, um Unterbrechungen besonders berücksichtigen
zu können, zum anderen muß - falls der Benutzer von der zuletzt gültigen abstrakten
Dialogstruktur abgewichen ist - das Benutzerwissen des Systems verändert werden.}
```

```
Procedure Bestimme_Akt_Arb_Zusammenhan (Abstr_Funk: Set of Abstr_Funktionen;
                                         Lay_Strukt: Layout_Struktur;
                                         Arb_Umgeb: Set of Arbeitszusammenhang;
                                         var: Akt_Arb_Zus: Arbeitszusammenhang);
{Es wird der aktuelle Arbeitszusammenhang bestimmt, um das weitere Vorgehen für
die Bearbeitung der Benutzereingabe vorzubereiten}
```

Begin {Verarbeitungsvorschrift_AKWE}

```
If Direkte Ausgabe ( Übergabemarke_akü12.Lay_Ein.Funktion,
                   Zustandsmarke_akzw.Layout_Strukt)
```

Then Begin

With Zustandsmarke_akzw **do**

Begin

```
Verändere_Abstrakte_Fenster (Realisierungsform,
                             Übergabemarke_akü12.Lay_Ein.Funktion,
                             Übergabemarke_akü12.Lay_Ein.Nachricht,
                             Layout_Strukt.Abstr_Fenster,
                             Ab_Funk1);
```

```
Verändere_Abstrakte_Einheiten (Realisierungsform,
                               Übergabemarke_akü12.Lay_Ein.Funktion,
                               Übergabemarke_akü12.Lay_Ein.Nachricht,
                               Layout_Strukt.Abstr_Einheiten,
                               Ab_Funk2);
```

```
Layout_Strukt.Akt_Funktion := Layout_Strukt.Akt_Funktion + Ab_Funk1 +
                             Ab_Funk2
```

{Die vor der Benutzereingabe gültigen Funktionen werden ergänzt bzw. teilweise ersetzt}

End;

With Übergabemarke_akü13 **do**

Begin

```
Log_Ein := 'undef';
```

{Bei der direkten Ausgabe wird keine logische Eingabe erstellt}

```
Layout_Strukt.Akt_Arbeitszusammenhang:=
```

```
    Zustandsmarke_akzw.Layout_Strukt.Akt_Arbeitszusammenhang;
```

```

Layout_Strukt.Akt_Abstr_Struktur:=
    Zustandsmarke_akzw.Layout_Strukt.Akt_Abstr_Struktur;
{Die Werte vor der Benutzeraktion behalten bei der direkten Ausgabe
weiterhin Gültigkeit.}
Layout_Strukt.Akt_Funktion:=
    Zustandsmarke_akzw.Layout_Strukt.Akt_Funktion;
Layout_Strukt.Abstr_Fenster:=
    Zustandsmarke_akzw.Layout_Strukt.Abstr_Fenster;
Layout_Strukt.Abstr_Einheiten:=
    Zustandsmarke_akzw.Layout_Strukt.Abstr_Einheiten;

    End
Else Begin
    Übergabemarke_akü13.Layout_Strukt := 'undef';
    {Für weitere Bearbeitungen der Benutzereingabe wird die Layout_Struktur
    nicht mehr benötigt}
    Bestimme_Akt_Dialogform (Übergabemarke_akü12.Lay_Ein.Funktion,
        Zustandsmarke_akzw.Layout_Strukt,
        Übergabemarke_akü12.Akt_Arbeitsumgebung,
        Übergabemarke_akü13.Log_Ein.Akt_Dialogform);
    Bestimme_Akt_Arb_Zusammenhang (Übergabemarke_akü12.Lay_Ein.Funktion,
        Zustandsmarke_akzw.Layout_Strukt,
        Übergabemarke_akü12.Akt_Arbeitsumgebung,
        Übergabemarke_akü13.Log_Ein.Akt_Arbeitszusammenhang);
    Bestimme_Log_Funktionen ( Übergabemarke_akü12.Lay_Ein.Funktion,
        Zustandsmarke_akzw.Layout_Strukt,
        Übergabemarke_akü13.Log_Ein.Funktion);
    Übergabemarke_akü13.Log_Ein.Nachricht:=
        Übergabemarke_akü12.Lay_Ein.Nachricht;

    End;
    Übergabemarke_akü13.Akt_Arbeitsumgebung:=
        Übergabemarke_akü12.Akt_Arbeitsumgebung
End;

```

Für die weitere Bearbeitung der ursprünglichen Benutzereingabe tritt bei dem Übergabekanal akü13 ein Konflikt zwischen mehreren möglichen Nachfolge-Instanzen ein. Wir lösen den Konflikt durch die

```

Procedure Konfliktlösungsvorschrift_akü13;
Begin
    If Übergabemarke_akü13.Log_Ein = 'undef'
    Then Verarbeitungsvorschrift_AKDA
    Else If Übergabemarke_akü13.Log_Ein.Akt_Dialogform = Unterbrechung
    Then Verarbeitungsvorschrift_AK1U
    Else Verarbeitungsvorschrift_AK1N;
End

```

da zunächst grundsätzlich zu unterscheiden ist, ob eine direkte Ausgabe erfolgt, oder nicht, und - falls nicht - die weitere Verarbeitung von der aktuellen Dialogform abhängt.

Für die direkte Ausgabe ist nun durch die Instanz AKDA lediglich eine Übergabe der Information des Eingangskanals zum Ausgangskanal notwendig, um die notwendigen geräteabhängigen Transformationen für die Ausgabe zu veranlassen.

```

Procedure Verarbeitungsvorschrift_AKDA;
  Begin
    With Übergabemarke_akü13 do
      Begin
        Übergabemarke_akü22.Layout_Strukt.Akt_Arbeitszusammenhang :=
          Layout_Strukt.Akt_Arbeitszusammenhang;
        Übergabemarke_akü22.Layout_Strukt.Akt_Funktion := Layout_Strukt.Akt_Funktion;
        Übergabemarke_akü22.Layout_Strukt.Akt_Abstr_Struktur :=
          Layout_Strukt.Akt_Abstr_Struktur;
        Übergabemarke_akü22.Layout_Strukt.Abstr_Fenster := Layout_Strukt.Abstr_Fenster;
        Übergabemarke_akü22.Layout_Strukt.Abstr_Einheiten := Layout_Strukt.Abstr_Einheiten;
        Übergabemarke_akü22.Akt_Arbeitsumgebung := Akt_Arbeitsumgebung
      End
    End

```

Ist durch die Benutzeraktion keine direkte Ausgabe gefordert, so muß durch die Instanzen AK1N oder AK1U die logische Eingabe vom Kanal akü13 weiter verarbeitet werden. Dabei ist im Falle einer Unterbrechung ein grundsätzlich anderes Vorgehen nötig als bei einer Datenabfrage, einem Kommando oder einem Auswahlangebot. Wir stellen die Diskussion der Verarbeitungsvorschrift AK1U vorerst zurück und erörtern zunächst die Funktionsweise von AK1N. Folgende Tätigkeiten sind durch die Instanz durchzuführen:

- Anpassung des Benutzerwissens bei akz1 auf der Grundlage der dort vermerkten abstrakten Dialogstruktur, der vom Benutzer gewählten Dialogform und des bisherigen Benutzerwissens; gegebenenfalls Setzen einer Funktion "Verändere abstrakte Dialogstruktur" zur Anpassung des Dialogverhaltens an den Benutzer;
- Reaktion auf die Benutzeraktion in Form von Anmerkungen, ohne die Bearbeitung der logischen Eingabe fortzusetzen (z.B. Fehlermeldungen, Nachfragen bei Benutzeraktionen mit schwerwiegenden Folgen);
- Feststellung des durch die Benutzeraktion angesprochenen Dialogpartners. Folgende Fälle sind zu berücksichtigen:
 1. die Systemreaktion erfolgt durch die logische Ebene (z.B. Verschieben von Fenstern, Ändern der Darstellungsart von Fenstern, Zeigen eines Pop-Up-Menüs);
 2. die Systemreaktion erfolgt durch den Arbeitsplatzrechner (z.B. Löschen von Daten, Darstellung von Text innerhalb eines Textsystems ändern);
 3. "keine" Systemreaktion, Weiterleiten der Benutzeraktion an benachbarte Benutzer;
 4. die Systemreaktion erfolgt durch eine in der Hierarchie nachgelagerte Komponente (z.B. Anmelden einer Anwendung auf dem Schulrechner);
 5. die Benutzeraktion fordert die Dialogbeendigung; hier muß eine Beendigung mit jeder Komponente der Hierarchie sichergestellt werden;

- Bearbeitung von Benutzeraktionen, die sich direkt auf die logische Ebene beziehen (siehe Punkt 1);
- die Schaffung der Möglichkeit einer Unterbrechung des Dialogs durch den Benutzer; hierfür setzen wir:

Type Übergabemarke_mu2 = (unterbrechbar, nicht_unterbrechbar)

Im Falle des Dialogendes und bei anderen Funktionen, vor deren Bearbeitung zunächst der Dialog mit einer nachgelagerten Systemkomponente beendet werden muß, ist auf akz2 eine Zwischenspeicherung der vom Benutzer ausgehenden Nachricht und der Funktionen notwendig. Um für die algorithmische Beschreibung die Übergabemarke akü24 nicht für solche Spezialfälle unnötig aufzublähen - die Zwischenspeicherung erfolgt bei AK4N, die zwischenspeichernde Information muß dorthin übergeben werden -, sehen wir eine logische Funktion "Zwischenspeicher" mit den beiden zu setzenden Parametern "Nachricht" vom Typ "Text" und "Funktion" vom Typ "Set of Logische_Funktionen" vor, die durch AK1N mit Werten belegt wird.

Im einzelnen stellt sich die Verarbeitungsvorschrift für AK1N - wieder mit nicht näher bezeichneten Prozeduren - wie folgt dar:

Procedure Verarbeitungsvorschrift_AK1N;

var Typ: Arbeitszusammenhang;
 L_Funk, L_F: **Set of** Logische_Funktionen;
 N1, N2: Text;
 Dia_Partner: Adressat;

Procedure Analysiere_Log_Eingabe (Lg_Funk: **Set of** Logische_Funktionen;
 Dia_Wissen: Log_Funktionsumfang;
 Var Lg_Reak_Funk: **Set of** Logische_Funktionen;
 Var Reak_Nachricht: Text);
 {Die Prozedur analysiert die logische Eingabe und stellt gegebenenfalls eine
 Reaktion als Anmerkung bzw. Fehlermeldung der logischen Ebene in Form von
 Funktionen und Nachrichten zur Verfügung}

Procedure Verändere_Benutzerwissen (Dia_Form: Dialogform;
 Dia_Strukt: Abstrakte_Dialogstruktur;
 Var Lg_Funk: **Set of** Logische_Funktionen;
 Var Ben_Wissen: Benutzermodell);
 {Mit Hilfe der abstrakten Dialogstruktur, der vom Benutzer gewählten Dialogform und
 der logischen Funktionen aus der logischen Eingabe verändert die Prozedur das Bild
 des Systems vom Benutzer und fügt gegebenenfalls die Funktion "Verändere_
 abstrakte_Dialogstruktur" zu der Menge logischer Funktionen hinzu.}

Procedure Ermittle_Dialogpartner
 (Lg_Funk: **Set of** Logische_Funktionen;
 Dia_Wissen: **Array**[Arbeitszusammenhang] **of** Log_Funktionsumfang;
 Var Partner: Adressat);
 {Die Prozedur stellt fest, welche Systemkomponente die logische Eingabe
 weiter zu bearbeiten hat}

Procedure Bearbeite_Aktion
 (Ben_Nachricht: Text;

```

    Lg_Ben_Funk: Set of Logische_Funktionen;
    Var Reak_Nachricht: Text;
    Var Lg_Reak_Funktion: Set of Logische_Funktionen);
    {Aus den durch die Benutzeraktion entstandenen logischen Funktionen und der vom
    Benutzer ausgehenden Nachricht ermittelt die Prozedur eine Systemreaktion in
    Form von logischen Funktionen und Nachrichten als an den Benutzer zu sendendes
    Bearbeitungsergebnis}
Procedure Zwischenspeicher (Lg_Funk: Set of Logische_Funktionen;
                           Nachricht: Text);
    {Die Prozedur belegt die Parameter der logischen Funktion
    "Zwischenspeicher" mit Werten.}

Begin
With Übergabemarke_akü13 do
    Begin
    If Log_Ein.Funktion = [Dialogende]
    Then If Zustandsmarke_akz1.Dialogrechner ≠ Arbeitsplatzrechner
        Then Begin
            Zwischenspeicher([Dialogende],Log_Ein.Nachricht);
            Übergabemarke_akü24.Funktion:=
                [Dialogende_Schulrechner, Zwischenspeicher];
            Übergabemarke_akü24.Dialogpartner:= Schulrechner
        End
        Else Übergabemarke_akü24.Dialogpartner:= Arbeitsplatzrechner
    Else Begin
        L_Funk:= [];
        N1:= '';
        For Typ := Kontrolle To AnwendungN Do
            Begin;
            Analysiere_Log_Eingabe    (Log_Ein.Funktion,
                                       Zustandsmarke_akz1.Dialogwissen[Typ],
                                       L_F,
                                       N2);

            L_Funk:= L_Funk + L_F;
            Concat (N1,N2)
            End;
        If L_Funk = []
        Then
            Begin {Die Analyse hat ergeben, daß keine Anmerkung als
                    Systemreaktion erfolgen muß}
            Ermittle_Dialogpartner    (Log_Ein.Funktion,
                                       Zustandsmarke_akz1.Dialogwissen
                                       [Log_Ein.Akt_Arbeitszusammenhang],
                                       Dia_Partner);

            Case Dia_Partner of
            Benutzer: Begin
                Übergabemarke_akü24.Dialogpartner := Benutzer;
                Bearbeite_Aktion    (Log_Ein.Nachricht,
                                     Log_Ein.Funktion,
                                     Übergabemarke_akü24.Funktion,
                                     Übergabemarke_akü24.Nachricht)
            End;
    End;

```

Arbeitsplatzrechner:

Begin

Zustandsmarke_akz1.Dialogpartner:= Arbeitsplatzrechner;

If Zustandsmarke_akz1.Dialogrechner ≠ Arbeitsplatzrechner

Then Begin

Zwischenspeicher(Log_Ein.Funktion,
Log_Ein.Nachricht);

Übergabemarke_akü24.Funktion:=

[Dialogende_Schulrechner, Zwischenspeicher];

Übergabemarke_akü24.Nachricht:= 'Vor

Bearbeitung der Funktion durch den Arbeits-
platzrechner ist eine Dialogbeendigung mit
dem Schulrechner erforderlich';

Übergabemarke_akü24.Dialogpartner:=Schulrechner

End

Else Begin

Übergabemarke_akü24.Funktion:=

Log_Ein.Funktion;

Übergabemarke_akü24.Nachricht:=

Log_Ein.Nachricht;

Übergabemarke_akü24.Dialogpartner:=

Arbeitsplatzrechner

End;

Zustandsmarke_akz1.Dialogrechner:= Arbeitsplatzrechner

End;

Nachbar1, ..., NachbarN:

Begin

Zustandsmarke_akz1.Dialogpartner:= Benutzer;

Übergabemarke_akü24.Funktion:= Log_Ein.Funktion;

Übergabemarke_akü24.Nachricht:= Log_Ein.Nachricht;

Übergabemarke_akü24.Dialogpartner:= Nachbar1, ...,
NachbarN

End;

{Zwar wird eine Nachricht an benachbarte Benutzer gesendet,
doch muß nach dem Absenden das System dem Benutzer in
unveränderter Arbeitsumgebung zur Weiterarbeit zu Verfügung
stehen (siehe Konfliktlösungsvorschrift_akü24)}

Schulrechner:

Begin

Zustandsmarke_akz1.Dialogpartner:= Schulrechner;

If Zustandsmarke_akz1.Dialogrechner ≠ Schulrechner

Then Zustandsmarke_akz1.Dialogrechner:=Schulrechner;

Übergabemarke_akü24.Funktion:= Log_Ein.Funktion;

Übergabemarke_akü24.Nachricht:= Log_Ein.Nachricht;

Übergabemarke_akü24.Dialogpartner:= Schulrechner

End;

{Eine eventuell nötige Dialogeröffnung mit dem
Schulrechner wird erst bei der Instanz AV1 veranlaßt}

End

End

Else

Begin {Die Analyse hat Anmerkungen als Systemreaktion ergeben}

```

Übergabemarke_akü24.Funktion:= L_Funk;
Übergabemarke_akü24.Nachricht:= N1;
Übergabemarke_akü24.Dialogpartner:= Benutzer;
Zustandsmarke_akz1.Dialogpartner:= Benutzer
End
End;
Übergabemarke_akü24.Akt_Arbeitszusammenhang:= Akt_Arbeitszusammenhang;
Übergabemarke_akü24.Akt_Arbeitsumgebung:= Akt_Arbeitsumgebung;
Verändere_Benutzerwissen( Akt_Arbeitszusammenhang,
                          Zustandsmarke_akz1.Akt_Abstr_Dialogstruktur,
                          Log_Ein.Funktion,
                          Zustandsmarke_akz1.Benutzerwissen);
Übergabemarke_mu2:= unterbrechbar
End
End {Verarbeitungsvorschrift_AK1N}

```

Bei der Übergabemarke akü24 tritt wiederum ein Konflikt zwischen mehreren möglichen Nachfolgern auf, für den eine Konfliktlösungsvorschrift anzugeben ist. Diese hängt aber wesentlich auch von der Funktionsweise der Instanz AK1U ab und wird von uns im nächsten Abschnitt in Zusammenhang mit der Diskussion der Verarbeitungsvorschrift AK1U angegeben.

Betrachten wir als nächstes die Instanz AK4N, die im Modell aus Abb. 13.6 Nachfolger von akü24 im Falle von normalen Dialogschritten mit dem Arbeitsplatzrechner oder einem nachgelagerten Rechner ist. Aufgabe dieser Instanz ist - falls der Arbeitsplatzrechner angesprochen ist - die Überführung der logischen Funktionen in die arbeitszusammenhangsabhängigen Bearbeitungsfunktionen. Ansonsten werden die Informationen lediglich zur Weiterbearbeitung übergeben. Um diese unterschiedlichen Arbeitsgänge und die aus ihnen jeweils resultierenden unterschiedlichen Strukturen berücksichtigen zu können, verwenden wir für die Übergabemarke_akü25 einen varianten Record:

```

Type   Dialogpartner = Adressat;
Übergabemarke_akü25 = Record
    Akt_Arbeitsumgebung: Set of Arbeitszusammenhang;
    Situation: (unverändert, neuer_Arbeitszusammenhang,
                alter_Arbeitszusammenhang);
    Form: Dialogformen;
    Case Dialogpartner of
    Arbeitsplatzrechner:
        (Bearb_Strukt: Bearbeitungsstruktur);
    Schulrechner, GroßRechner:
        (Funktion: Set of Logische_Funktionen;
         Nachricht: Text;
         Akt_Arbeitszusammenhang: Arbeitszusammenhang)
End

```

Für die Bearbeitung der Information des Eingangskanals muß unterschieden werden, ob der Arbeitsplatzrechner oder eine nachgelagerte Systemkomponente

zuständig ist. Im Falle des Arbeitsplatzrechners ist es weiter erforderlich, an dieser Stelle die Situationen zu berücksichtigen,

- daß durch die zu bearbeitende Information der zuletzt gültige aktuelle Arbeitszusammenhang weiterhin Bestand haben kann,
- daß ein neuer Arbeitszusammenhang festgelegt wurde, der zuvor nicht in der aktuellen Arbeitsumgebung enthalten war, oder
- daß der Wechsel zu einem Arbeitszusammenhang vorliegt, der bereits bekannt, in früheren Dialogschritten bereits bearbeitet und dann unterbrochen wurde.

Je nach Situation sind unterschiedliche Vorgehensweisen notwendig, bzw. Folgeinstanzen zu bestimmen. Wir beschreiben die Funktionsweise der Instanz AK4N wieder mit Hilfe nicht näher spezifizierter Prozeduren durch:

Procedure Verarbeitungsvorschrift_AK4N;

Type Speicher = **Record**

Log_Funk: **Set of** Logische_Funktionen;

Nachricht: Text

End;

Procedure Fülle_Zwischenspeicher (Lg_Funk: Logische_Funktionen;

Var Zw_Speicher: Speicher);

{Durch die Prozedur wird die zunächst zwischenzuspeichernde Information in den Zwischenspeicher der Zustandsmarke akz2 eingetragen, wenn vor Bearbeitung der Information andere Schritte zur Vorbereitung ausgeführt werden müssen.}

Procedure Bestimme_Bearbeitungsfunktionen

(Lg_Funk: **Set of** Logische_Funktionen;

A_Wissen: **Array**{Arbeitszusammenhang} **of** Bearb_Funktionsumfang;

Var Bearb_Funk: **Set of** Bearbeitungsfunktionen);

{Mit Hilfe des Anwendungswissens werden die logischen Funktionen in die sie darstellenden Bearbeitungsfunktionen überführt}

Begin {Verarbeitungsvorschrift_AK4N}

With Zustandsmarke_akz2 **Do**

Begin

Zwischenspeicher.Funktion := [];

Zwischenspeicher.Nachricht:= '';

Dialogpartner:= Übergabemarke_akü24.Dialogpartner;

Dialogrechner:= Übergabemarke_akü24.Dialogpartner

End;

With Übergabemarke_akü24 **Do**

If [Zwischenspeicher] **In** Funktion

Then Begin

Fülle_Zwischenspeicher(Zwischenspeicher,

Zustandsmarke_akz2.Zwischenspeicher);

Funktion:= Funktion - [Zwischenspeicher]

End;

{Im Bedarfsfall wird der Zwischenspeicher gefüllt und die zu bearbeitenden Funktionen um die "Zwischenspeicher"-Funktion reduziert.}

With Übergabemarke_akü25 **Do**

Begin

 Dialogpartner:= Übergabemarke_akü24.Dialogpartner;

If Dialogpartner = Arbeitsplatzrechner

Then Begin {Die logische Form wird in die Bearbeitungsstruktur überführt}

 Bearb_Strukt.Nachricht:= Übergabemarke_akü24.Nachricht;

 Bearb_Strukt.Akt_Arbeitszusammenhang:=

 Übergabemarke_akü24.Akt_Arbeitszusammenhang;

If Bearb_Strukt.Akt_Arbeitszusammenhang ≠

 Zustandsmarke_akz2.Akt_Arbeitszusammenhang

Then {Der zuletzt gültige Arbeitszusammenhang wurde verlassen}

If Bearb_Strukt.Akt_Arbeitszusammenhang **In**

 Übergabemarke_akü24.Akt_Arbeitsumgebung

Then {ein zuvor schon bearbeiteter Arbeitszusammenhang wurde gewählt}

Begin

 Situation:= alter_Arbeitszusammenhang;

 Akt_Arbeitsumgebung:=

 Übergabemarke_akü24.Akt_Arbeitsumgebung

End

Else {ein zuvor noch nicht bearbeiteter Arbeitszusammenhang wurde gewählt}

Begin

 Situation:= neuer_Arbeitszusammenhang;

 Akt_Arbeitsumgebung:=

 Übergabemarke_akü24.Akt_Arbeitsumgebung

 + Übergabemarke_akü24.Akt_Arbeitszusammenhang

End

Else {der zuletzt gültige Arbeitszusammenhang bleibt bestehen}

Begin

 Situation:= unverändert;

 Akt_Arbeitsumgebung:= Übergabemarke_akü24.Akt_Arbeitsumgebung

End;

 Bestimme_Bearbeitungsfunktionen(Übergabemarke_akü24.Funktion,
 Zustandsmarke_akz2.Anwendungswissen,
 Bearb_Strukt.Funktion)

End

Else Begin {Die Information der logischen Ebene wird nur übergeben, der Aufbau der Bearbeitungsstruktur erfolgt erst bei SK1N}

 Funktion:= Übergabemarke_akü24.Funktion;

 Nachricht:= Übergabemarke_akü24.Nachricht;

 Akt_Arbeitszusammenhang:=

 Übergabemarke_akü24.Akt_Arbeitszusammenhang

End

End

End {Verarbeitungsvorschrift_AK4N}

Anhand des gesetzten Dialogpartners und der vorliegenden Situation wird der bei akü25 vorliegende Konflikt gemäß der folgenden Vorschrift für eine der Instanzen AU, AN1, AW oder AV1 entschieden:

```

Procedure Konfliktlösungsvorschrift_akü25;
Begin
  With Übergabemarke_akü25 do
    If Dialogpartner = Schulrechner
      Then If Form = Unterbrechung
        Then Cobegin
          Verarbeitungsvorschrift_AU;
          Verarbeitungsvorschrift_AV1
        Coend
        Else Verarbeitungsvorschrift_AV1
      Else If [Unterbrechung] In Bearb_Strukt.Funktion
        Then Verarbeitungsvorschrift_AU
      Else If Situation = alter_Arbeitszusammenhang
        Then Verarbeitungsvorschrift_AW
        Else Verarbeitungsvorschrift_AN1
  End

```

Die eigentliche Bearbeitung der Benutzereingabe durch den Arbeitsplatzrechner erfolgt durch die Instanzen AN1 bis AN4, für deren Funktion wir davon ausgehen wollen, daß sie jeweils auf der Grundlage einer Bearbeitungsstruktur bestehend aus Funktionen und Nachrichten neue Funktionen und neue Nachrichten, also eine neue, veränderte Bearbeitungsstruktur, ermitteln. Es ist dann mit

```

Type Übergabemarke_au1 = (unterbrechbar);
  Übergabemarke_az1 = Record
    Bearb_Strukt: Bearbeitungsstruktur;
    Dialogpartner: Adressat;
    Dialogrechner: Hierarchie;
    Akt_Arbeitsumgebung: Set of Arbeitszusammenhang
  End

```

ihre Funktionsweise allgemein zu beschreiben durch:

```

Procedure Verarbeitungsvorschrift_AN1;
  Function Partner: Dialogpartner;
    {Bestimmung des nach Bearbeitung der Eingabe
    zuständigen Dialogpartners. In der Regel der Benutzer.}
  Procedure Verändere_Bearb_Struktur (B_Alt_Strukt: Bearbeitungsstruktur
    Var B_Neu_Strukt: Bearbeitungsstruktur);
    { Aus der Eingangs-Bearbeitungsstruktur wird entsprechend des
    Arbeitszusammenhangs eine neue Bearbeitungsstruktur ermittelt}

  Begin
    Verändere_Bearb_Struktur (Übergabemarke_akü25.Bearb_Strukt,
      Übergabemarke_az1.Bearb_Strukt);
    Übergabemarke_az1.Dialogpartner:= Partner;
    Übergabemarke_az1.Dialogrechner:= Arbeitsplatzrechner;
    {Festhalten, daß der Arbeitsplatzrechner aktiv ist.}
    Übergabemarke_az1.Akt_Arbeitsumgebung:=Übergabemarke_akü25.Akt_Arbeitsumgebung;
    Übergabemarke_au1:= unterbrechbar
  End

```

Die Algorithmen für AN2, AN3 und AN4 folgend entsprechend, wobei az2 und az3 wie az1 strukturiert sind und AN4 zusätzlich bei dem Zustandskanal az die Parameter setzt. Die Struktur des Kanals akü14 wird weiter unten beschrieben. Bei dem Zustandskanal az wird unter Anwendungsinhalt für jeden in der Arbeitsumgebung gültigen Arbeitszusammenhang die jeweils letzte Bearbeitungsstruktur vermerkt. Dies ermöglicht das Wiederaufsetzen, d.h. erneute Aktivieren eines Arbeitszusammenhanges zu beliebigem Zeitpunkt. Wenn also die Konfliktlösungsvorschrift_akü25 die Instanz AW als Nachfolger festlegt, so wird durch diese die zuletzt gültige Bearbeitungsstruktur zu dem aktuellen Typ ausgewählt und "geladen". Algorithmisch zu beschreiben ist dies unter Verwendung von

```
Type Übergabemarke_awü = Record
    Bearb_Strukt: Bearbeitungsstruktur;
    Dialogpartner: Adressat;
    Akt_Arbeitsumgebung: Set of Arbeitszusammenhang;
    Zustand: 1 .. 4
End
```

durch folgende Vorschrift:

```
Procedure Verarbeitungsvorschrift_AW;
Begin
    With Übergabemarke_awü Do
        Begin Bearb_Strukt :=
            Anwendungsinhalt[Zustandsmarke_az.Akt_Arbeitszusammenhang].Inhalt;
            Dialogpartner :=
                Anwendungsinhalt[Zustandsmarke_az.Akt_Arbeitszusammenhang].Dialogpartner;
            Zustand :=
                Anwendungsinhalt[Zustandsmarke_az.Akt_Arbeitszusammenhang].Zustand;
            Akt_Arbeitsumgebung := Übergabemarke_akü25.Akt_Arbeitsumgebung
        End
    End
```

Die daraufhin bei awü vorzufindende Konfliktsituation wird unter Verwendung von Übergabemarke_awü.Zustand für eine der Instanzen AWN1 bis AWN4 entschieden, die ihrerseits dann die Bearbeitungsinformation aus dem Kanal awü auf ihren jeweiligen Ausgangskanal übertragen. Wir wollen die Prozeduren im einzelnen nicht angeben, sondern stattdessen auf die Funktionsweise der Instanz AK2 eingehen.

Voraussetzung für die Arbeit von AK2 ist die Markierung des Kanals akü14. Dessen Struktur ist in einer Variante gleich der von az1, wenn nämlich der Arbeitsplatzrechner den Kanal beschrieben hat. In einer zweiten Variante enthält akü14 Informationen auf der logischen Ebene, erstellt durch die Instanz SK2 des Schulrechners, bzw. GK2 des Großrechners. Zusätzlich ist auch hier die Komponente Form aufgenommen worden, um den Konflikt zwischen AK2 und AUE zu lösen (siehe nächsten Abschnitt).

```

Type Dialogrechner= Hierarchie;
Übergabemarke_akü14 = Record
    Akt_Arbeitsumgebung: Set of Arbeitszusammenhang;
    Form: Dialogformen;
    Dialogpartner: Adressat;
Case Dialogrechner of
    Arbeitsplatzrechner
        (Bearb_Strukt: Bearbeitungsstruktur);
    Schulrechner, Großrechner:
        (Funktion: Set of Logische_Funktionen;
        Nachricht: Text;
        Akt_Arbeitszusammenhang: Arbeitszusammenhang)
End

```

Weiter ist durch AK2 zu unterscheiden, ob in den Bearbeitungsschritten zuvor die eigentliche Eingabe behandelt wurde, oder ob es sich um eine notwendig gewordene zwischengeschobene Bearbeitung gehandelt hat (Beendigung des Schulrechner_Dialogs vor Ausführung von Funktionen, die sich auf den Arbeitsplatzrechner beziehen). Im zweiten Fall befinden sich eine Menge logischer Funktionen und eine Nachricht im Zwischenspeicher von akz2, die durch AK2 "geladen" werden müssen. Im ersten Fall ist die Bearbeitungsstruktur von akü14 in die logische Form zu transformieren, bzw. die Information der logischen Ebene an akü24 zu übergeben.

```

Procedure Verarbeitungsvorschrift_AK2;
Procedure Bestimme_Log_Funktionen
    (Bearb_Funk: Set of Bearbeitungsfunktionen;
    A_Wissen: Array[Arbeitszusammenhang] of Bearb_Funktionsumfang;
    Var Log_Funk: Set of Logische_Funktionen);
    {Zu einer Menge von Bearbeitungsfunktionen wird die sie darstellende Menge
    logischer Funktionen gebildet}

Begin {Verarbeitungsvorschrift_AK2}
With Zustandsmarke_akz2 do
    Begin
    Dialogpartner:= Übergabemarke_akü14.Dialogpartner;
    If Übergabemarke_akü14.Dialogrechner = Arbeitsplatzrechner
        Then Akt_Arbeitszusammenhang:=
            Übergabemarke_akü14.Bearb_Strukt.Akt_Arbeitszusammenhang
        Else Akt_Arbeitszusammenhang :=
            Übergabemarke_akü14.Akt_Arbeitszusammenhang
    {Dialogpartner und Dialogrechner werden von den nachgelagerten Instanzen bestimmt
    Im Falle der Dialogbeendigung mit dem Schulrechner wird von SB als Dialogpartner der
    Arbeitsplatzrechner gesetzt}
    End;
With Übergabemarke_akü24 Do
    Begin
    If Zustandsmarke_akz2.Zwischenspeicher.Funktion ≠ []
        Then Begin {Zwischenspeicher ist gefüllt}
            Funktion:= Zustandsmarke_akz2.Zwischenspeicher.Funktion;
            Nachricht:= Zustandsmarke_akz2.Zwischenspeicher.Nachricht;

```

```

Zustandsmarke_akz2.Dialogrechner:= Arbeitsplatzrechner
End
Else {Zwischenspeicher ist leer}
  If Übergabemarke_akü14.Dialogrechner = Arbeitsplatzrechner
    Then Begin
      Bestimme_Log_Funktionen
      (Übergabemarke_akü14.Bearb_Strukt.Funktion,
       Zustandsmarke_akz2.Anwendungswissen,
       Funktion);
      Nachricht:= Übergabemarke_akü14.Bearb_Strukt.Nachricht;
      Akt_Arbeitszusammenhang:=
        Übergabemarke_akü14.Bearb_Strukt.Akt_Arbeitszusammenhang;
    End
  Else Begin
    Funktion:= Übergabemarke_akü14.Funktion;
    Nachricht:= Übergabemarke_akü14.Nachricht;
    Akt_Arbeitszusammenhang:=
      Übergabemarke_akü14.Akt_Arbeitszusammenhang
  End;
  Dialogpartner:= Übergabemarke_akü14.Dialogpartner;
  Akt_Arbeitsumgebung:= Übergabemarke_akü14.Akt_Arbeitsumgebung
End
End {Verarbeitungsvorschrift_AK2}

```

Ist auf der Übergabemarke akü25 als Dialogpartner der Schulrechner gesetzt, so wird durch die Konfliktlösungsvorschrift_akü25 die Instanz AV1 als Nachfolger gesetzt. Diese kann durch den Wert der Komponente "Dialogrechner" der Zustandsmarke az feststellen, ob der Dialog mit dem Schulrechner bereits besteht oder zunächst zu eröffnen ist. Im ersten Fall übergibt sie die zu bearbeitende Information an den Kanal skü1, im zweiten wird die Menge der Funktionen um die "Dialogeröffnungs"- Funktion ergänzt und dann die Information übergeben. Mit

```

Type Übergabemarke_skü1 = Record
  Dialogpartner: Adressat;
  Akt_Arbeitsumgebung: Set of Arbeitszusammenhang;
  Funktion: Set of Logische_Funktionen;
  Nachricht: Text;
  Akt_Arbeitszusammenhang: Arbeitszusammenhang
End

```

gilt also:

```

Procedure Verarbeitungsvorschrift_AV1;
  Var L_Funk: Set of Logische_Funktionen;
  Begin
    L_Funk:= [];
    If Zustandsmarke_az.Dialogrechner ≠ Schulrechner
      Then Begin
        Zustandsmarke_az.Dialogrechner:= Schulrechner;

```

```

        L_Funk:= [Dialogeroeffnung_Schulrechner]
    End;
With Übergabemarke_skü1 Do
    Begin
        Funktion:= Übergabemarke_akü25.Funktion + L_Funk;
        Nachricht:= Übergabemarke_akü25.Nachricht;
        Akt_Arbeitszusammenhang:= Übergabemarke_akü25.Akt_Arbeitszusammenhang;
        Dialogpartner:= Übergabemarke_akü25.Dialogpartner;
        Akt_Arbeitsumgebung:= Übergabemarke_akü25.Akt_Arbeitsumgebung
    End
End

```

Der bei skü1 auftretende Konflikt zwischen den Instanzen SK1N, SK1U und SE wird durch Betrachtung der bei skü1 definierten Funktionen gelöst:

```

Procedure Konfliktlösungsvorschrift_skü1;
Begin
    With Übergabemarke_skü1 Do
        If [Dialogeroeffnung_Schulrechner] In Funktion
            Then Verarbeitungsvorschrift_SE
        Else If [Unterbrechung] In Funktion
            Then Verarbeitungsvorschrift_SK1U
            Else Verarbeitungsvorschrift_SK1N
    End

```

Wir wollen an dieser Stelle die Diskussion der Attributsmarken und Instanzenvorschriften im Rahmen des normalen Dialogs abbrechen, weil die Funktionsweise der Instanzen des Schul- und des Großrechners nichts prinzipiell Neues gegenüber den bisher besprochenen Instanzen enthalten:

- SK1N und GK1N analysieren die Eingabe (in der Weise, wie AK1N), d.h. stellen zunächst fest, ob der durch sie repräsentierte Rechner angesprochen ist und ob gegebenenfalls vor Bearbeitung der Funktionen Rückfragen notwendig sind. Weiter stellen sie die Verbindung der logischen Ebene zur Bearbeitungsebene dar und überführen - wie beim Arbeitsplatzrechner AK4N - die Information in die Bearbeitungsstruktur. Ebenfalls wie AK4N stellen sie eine Situation für die Weiterbearbeitung fest. SK1N verwaltet zusätzlich einen Zwischenspeicher, falls zwischengeschobene Funktionen auf dem Großrechner erledigt werden müssen, bevor der Schulrechner mit der Bearbeitung der eigentlichen Eingabe beginnt. (Funktionsweise wie AK4N zum Zwischenspeichern und AK2 zum Zurückholen).
- SE und GE initialisieren die Zustandskanäle (wie AE).
- SN1, ..., SN4, GN1, ..., GN4 bearbeiten und erzeugen Bearbeitungsstrukturen (zu beschreiben wie für AN1, ..., AN4).
- SW und GW, sowie die ihnen nachfolgenden Instanzen setzen bei vorhergehenden Arbeitszusammenhängen auf (in der gleichen Weise, wie für AW beschrieben).
- SK2 und GK2 schließlich erfüllen jeweils auf ihrer Hierarchieebene die gleiche

Funktion, wie beim Arbeitsplatzrechner die Instanz AK2, transformieren also Information von der Bearbeitungsebene in die logische Ebene.

14.4. Algorithmen und Typen zur Beschreibung von Unterbrechungen

Eine Unterbrechung des "normalen" Dialogablaufs zwischen einem Benutzer und dem Rechnersystem geht immer vom Benutzer aus, im Modell aus Abb. 13.6. durch die Instanz MU dargestellt. Die Unterbrechung erreicht als physikalische Funktion die Instanz AKGE und nach Transformation in die Layout-Ebene die Instanz AKWE. Bis zu diesem Zeitpunkt wird sie auf der "Eingabe-Pipeline" wie alle anderen Benutzer-Aktionen behandelt. Bei AKWE wird nun die Dialogform "Unterbrechung" erkannt, woraufhin die Konfliktlösungsvorschrift für akü13 die Instanz AK1U zur Bearbeitung der Benutzereingabe festlegt.

Die Unterbrechung kann in zwei Ausprägungen auftreten. Zum einen ist möglich, daß der Benutzer den normalen Ablauf "einfach nur unterbricht", d.h. abbricht. Andererseits kann er "gleichzeitig" mit dem Unterbrechen andere Funktionen eingegeben haben, z.B. Auswahl eines neuen Arbeitszusammenhangs, um dort unmittelbar den normalen Dialog zu beginnen, bzw. fortzusetzen. (Realisiert wird der Wechsel des Arbeitszusammenhangs z.B. durch "Anklicken" eines Fensters auf dem Bildschirm mit der Maus.)

Um diese zweite Ausprägung einer Unterbrechung "benutzerfreundlich" modellieren zu können, ist von einer eingegebenen Funktionenmenge bei der Komponente Log_Ein der Marke akü13 auszugehen. (Als Alternative wäre denkbar, daß in einem Dialogschritt unterbrochen wird, das System daraufhin dem Benutzer die Eingabeberechtigung erteilt und dieser dann im nächsten Dialogschritt den Ziel-Arbeitszusammenhang bestimmt: eine Form, die z.B. im BS3 der TR 440 realisiert wurde. Das ist aber gerade nicht benutzerfreundlich.)

Bezüglich der neben der "Unterbrechungsfunktion" in Log_Ein der Marke akü13 enthaltenen Funktionen ist die Fallunterscheidung zur Verarbeitungsvorschrift AK1N zu berücksichtigen: Die Funktionen können sich auf die logische Ebene beziehen, das Dialogende herbeiführen oder einen Rechner innerhalb der Hierarchie betreffen. Weiter ist zu unterscheiden, in welchem Bearbeitungsstadium der zu unterbrechende Ablauf sich gerade befindet: Ist einer der Rechner noch aktiv oder ist bereits eine Ausgabe durch AK3N generiert. Folgende Konstellationen sind denkbar:

1. Der zu unterbrechende Ablauf befindet sich im Ausgabestadium, und keine weitere Funktion wurde ausgelöst. Dann ist eine Unterbrechungsnachricht durch AK3U zu generieren (Erteilen der Eingabeberechtigung an den Benutzer) und die vorangegangene Ausgabe zu unterdrücken.
2. Der zu unterbrechende Ablauf befindet sich im Ausgabestadium, und die

weiteren Funktionen beziehen sich auf die logische Ebene. Dann ist außer der Ausgabeunterdrückung eine Bearbeitung dieser Funktionen von AK1U durchzuführen und das Ergebnis über AK3N an den Benutzer zu leiten.

3. Der zu unterbrechende Ablauf befindet sich im Ausgabestadium, und die weiteren Funktionen beziehen sich auf einen Rechner der Hierarchie. Dann ist außer der Ausgabeunterdrückung eine weitere Bearbeitung der Funktionen durch die Instanz AW, SW oder GW notwendig.
4. Der zu unterbrechende Ablauf befindet sich im Ausgabestadium, und die weiteren Funktionen richten sich an einen benachbarten Benutzer. Dann ist außer der Ausgabeunterdrückung eine Datenübertragung mittels der Instanz DÜ einzuleiten.
5. Der zu unterbrechende Ablauf befindet sich im Ausgabestadium, und die weiteren Funktionen fordern das Dialogende. Dann ist der Ausgabevorgang abzubrechen und das Dialogende durch AB (gegebenenfalls auch SB und GB) herbeizuführen.
6. Der zu unterbrechende Ablauf befindet sich in Bearbeitung durch einen der Rechner der Hierarchie, sonst wie 1. Dann ist die Bearbeitung mittels der Instanz AU (bzw. SU oder GU) zu beenden und eine Meldung an den Benutzer zu senden.
7. Der zu unterbrechende Ablauf befindet sich in Bearbeitung durch einen der Rechner der Hierarchie, sonst wie 2. Dann ist außer der Bearbeitungsunterbrechung mittels AU eine Bearbeitung der Funktionen von AK1U durchzuführen und das Ergebnis über AK3N an den Benutzer zu senden.
8. Der zu unterbrechende Ablauf befindet sich in Bearbeitung durch einen der Rechner der Hierarchie, sonst wie 3. Dann müssen die Funktionen zwischengespeichert werden, der vorangegangene Ablauf durch AU (bzw. SU oder GU) gestoppt und dann die Bearbeitung der zwischengespeicherten Funktionen aufgenommen werden.
9. Der zu unterbrechende Ablauf befindet sich in Bearbeitung durch einen der Rechner der Hierarchie, sonst wie 4. Dann ist eine Vorgehensweise entsprechend Fall 8. möglich.
10. Der zu unterbrechende Ablauf befindet sich in Bearbeitung durch einen der Rechner der Hierarchie, sonst wie 5. Ebenfalls ist eine Vorgehensweise entsprechend Fall 8. möglich.

Beschreiben wir zunächst den Ablauf zur Unterdrückung der Ausgabe eines vorangegangenen Dialogschritts:

Durch die Unterbrechungsinstanz MU wird die Übergabemarke mu1 mit einem Wert belegt und dadurch gerade diejenige Instanz AKAU, AKGU, AKWU, AKÜU oder AKUE schaltfähig, die auf ihrem zweiten Eingangskanal ebenfalls eine Marke findet. Wir setzen

Type Übergabemarke_mu1 = (unterbrochen);

```

Procedure Verarbeitungsvorschrift_AKÜU;
  Begin
    Übergabemarke_mu1 := 'undef';
    Übergabemarke_akü23 := 'undef'
  End

```

und interpretieren das Setzen von 'undef' als Vernichten der auf diesem Kanal befindlichen Information. Die Verarbeitungsvorschriften für AKAU, AKGU, AKWU und AKUE folgen entsprechend.

Für die Funktionsweise der Instanz AK1U müssen aufgrund der obigen Überlegungen zunächst - sofern vorhanden - die zusätzlichen Funktionen analysiert und der für sie zuständige Dialogpartner gefunden werden. Danach müssen je nach Ablaufstadium Funktionen zur Ansteuerung der zuständigen Ziel-Instanzen gesetzt werden. In Anlehnung an AK1N setzen wir:

```

Procedure Verarbeitungsvorschrift_AK1U;
  var Typ: Arbeitszusammenhang;
      L_Funk: Set of Logische_Funktionen;
      Dia_Partner: Adressat;

  Procedure Verändere Benutzerwissen (Dia_Form: Dialogform;
                                     Dia_Strukt: Abstrakte_Dialogstruktur;
                                     Var Lg_Funk: Set of Logische_Funktionen;
                                     Var Ben_Wissen: Benutzermodell);
    {Mit Hilfe der abstrakten Dialogstruktur, der vom Benutzer gewählten Dialogform und
     der logischen Funktionen werden das Bild des Systems vom Benutzer verändert
     und gegebenenfalls die Funktion"Verändere_abstrakte_Dialogstruktur" zu der
     Menge logischer Funktionen hinzugefügt.}

  Procedure Ermittle_Dialogpartner
    (Lg_Funk: Set of Logische_Funktionen;
     Dia_Wissen: Array[Arbeitszusammenhang] of Log_Funktionsumfang;
     Var Partner: Adressat);
    {Die Prozedur stellt fest, welche Systemkomponente die logische Eingabe weiter zu
     bearbeiten hat}

  Procedure Bearbeite_Aktion(Ben_Nachricht: Text;
                             Lg_Ben_Funk: Set of Logische_Funktionen;
                             Var Reak_Nachricht: Text;
                             Var Lg_Reak_Funktion: Set of Logische_Funktionen);
    {Aus den durch die Benutzeraktion entstandenen logischen Funktionen und der
     vom Benutzer ausgehenden Nachricht ermittelt die Prozedur eine Systemreaktion
     in Form von logischen Funktionen und Nachrichten als an den Benutzer zu
     sendendes Bearbeitungsergebnis}

  Procedure Zwischenspeicher (Lg_Funk: Set of Logische_Funktionen;
                              Nachricht: Text);
    {Die Prozedur belegt die Parameter der logischen Funktion "Zwischenspeicher" mit
     Werten.}

  Begin {Verarbeitungsvorschrift_AK1U}
  With Übergabemarke_akü13 do
    Begin
      L_Funk:= Log_Ein.Funktion - {Unterbrechung};

```

```

If L_Funk ≠ []
Then Begin {Es sind zusätzliche Funktionen vorhanden}
    Ermittle_Dialogpartner (L_Funk,
        Zustandsmarke_akz1.Dialogwissen
        [Log_Ein.Akt_Arbeitszusammenhang],
        Dia_Partner);

    Case Dia_Partner of
Benutzer:
    Begin {Fall 2 oder Fall 7}
        Übergabemarke_akü24.Dialogpartner := Benutzer;
        Bearbeite_Aktion (Log_Ein.Nachricht,
            L_Funk,
            Übergabemarke_akü24.Funktion,
            Übergabemarke_akü24.Nachricht);
    If Zustandsmarke_akz1.Dialogpartner ≠ Benutzer
    Then Begin {Fall 7}
        Zustandsmarke_akz1.Dialogpartner:= Benutzer;
        Übergabemarke_akü24.Funktion:=
        Übergabemarke_akü24.Funktion + [Unterbrechung]
    End
    End;
Arbeitsplatzrechner:
    Begin {Fall 3, 5, 8 oder 10}
    If Zustandsmarke_akz1.Dialogrechner ≠ Arbeitsplatzrechner
    Then Begin
        Zwischenspeicher(L_Funkt, Log_Ein.Nachricht);
        Übergabemarke_akü24.Funktion:=
        [Dialogende_Schulrechner, Zwischenspeicher];
        Übergabemarke_akü24.Nachricht:= 'Vor Bearbeitung der
        Funktion durch den Arbeitsplatzrechner ist eine
        Dialogbeendigung mit dem Schulrechner erforderlich';
        Übergabemarke_akü24.Dialogpartner:= Schulrechner;
        If Zustandsmarke_akz1.Dialogpartner ≠ Benutzer
        Then {Fall 8 oder Fall 10}
            Übergabemarke_akü24.Funktion:=
            Übergabemarke_akü24.Funktion + [Unterbrechung]
        End
    End
    Else Begin
        If Zustandsmarke_akz1.Dialogpartner ≠ Benutzer
        Then Begin {Fall 8 oder Fall 10}
            Zwischenspeicher(L_Funk, Log_Ein.Nachricht);
            Übergabemarke_akü24.Funktion:= [Unterbrechung];
            Übergabemarke_akü24.Nachricht := ' Nur
            Unterbrechen, die eigentlichen Funktionen
            sind zwischengespeichert '
        End
        Else Begin
            Übergabemarke_akü24.Funktion:= L_Funk;
            Übergabemarke_akü24.Nachricht:=Log_Ein.Nachricht
        End
        Übergabemarke_akü24.Dialogpartner:=Arbeitsplatzrechner
    End;

```

```

Zustandsmarke_akz1.Dialogpartner:= Arbeitsplatzrechner;
Zustandsmarke_akz1.Dialogrechner:= Arbeitsplatzrechner
End
Nachbar1, ..., NachbarN:
  Begin {Fall 4 oder Fall 9}
    Übergabemarke_akü24.Funktion:= L_Funk;
    Übergabemarke_akü24.Nachricht:= Log_Ein.Nachricht;
    Übergabemarke_akü24.Dialogpartner:= Nachbar1, ..., NachbarN;
    If Zustandsmarke_akz1.Dialogpartner ≠ Benutzer
      Then Begin {Fall 9}
        Zustandsmarke_akz1.Dialogpartner:= Benutzer;
        Übergabemarke_akü24.Funktion:=
          Übergabemarke_akü24.Funktion + [Unterbrechung]
      End
    End;
Schulrechner:
  Begin {Fall 3, 5, 8 oder 10}
    Übergabemarke_akü24.Funktion:= L_Funk;
    Übergabemarke_akü24.Nachricht:= Log_Ein.Nachricht;
    Übergabemarke_akü24.Dialogpartner:= Schulrechner
    If Zustandsmarke_akz1.Dialogpartner ≠ Benutzer
      Then {Fall 8 oder Fall 10}
        Übergabemarke_akü24.Funktion:=
          Übergabemarke_akü24.Funktion + [Unterbrechung]
        Zustandsmarke_akz1.Dialogpartner:= Schulrechner;
        Zustandsmarke_akz1.Dialogrechner:= Schulrechner
      End
    End
Else Begin {Keine zusätzlichen Funktionen: Fall 1 oder Fall 6}
  Übergabemarke_akü24.Dialogpartner:= Benutzer;
  Übergabemarke_akü24.Funktion:= [Unterbrechungsmeldung];
  Übergabemarke_akü24.Nachricht:= 'Eingabeberechtigung erteilen';
  If Zustandsmarke_akz1.Dialogpartner ≠ Benutzer
    Then Begin {Fall 8 oder Fall 10}
      Zustandsmarke_akz1.Dialogpartner:= Benutzer;
      Übergabemarke_akü24.Funktion:=
        Übergabemarke_akü24.Funktion + [Unterbrechung]
    End
  End
  Übergabemarke_akü24.Akt_Arbeitszusammenhang:= Akt_Arbeitszusammenhang;
  Übergabemarke_akü24.Akt_Arbeitsumgebung:= Akt_Arbeitsumgebung;
  Verändere_Benutzerwissen ( Akt_Arbeitszusammenhang,
    Zustandsmarke_akz1.Akt_Abstr_Dialogstruktur,
    Log_Ein.Funktion,
    Zustandsmarke_akz1.Benutzerwissen);
End
End {Verarbeitungsvorschrift_AK1U}

```

Zur Lösung des bei akü24 auftretenden Konflikts setzen wir die Konfliktlösungsvorschrift in Form folgender Prozedur:

```

Procedure Konfliktlösungsvorschrift_akü24;
Begin
  With Übergabemarke_akü24 do
    If [Dialogende] in Funktion
      Then Verarbeitungsvorschrift_AB {normales Beenden}
      Else Case Dialogpartner of
        Benutzer:
          If [Unterbrechung] in Funktion
            Then If [Unterbrechungsmeldung] in Funktion
              Then Cobegin {Fall 6}
                Verarbeitungsvorschrift_AK3U;
                Verarbeitungsvorschrift_AKUE;
                Verarbeitungsvorschrift_AK4U
              Coend
              Else Cobegin {Fall 7}
                Verarbeitungsvorschrift_AK3N;
                Verarbeitungsvorschrift_AK4U,
                Verarbeitungsvorschrift_AKUE
              Coend
            Else If [Unterbrechungsmeldung] in Funktion
              Then Verarbeitungsvorschrift_AK3U {Fall 1}
              Else Verarbeitungsvorschrift_AK3N; {Fall 2, normaler Dialog}
        Nachbar1, ..., NachbarN:
          If [Unterbrechung] in Funktion
            Then Cobegin {Fall 9}
              «Verarbeitungsvorschrift_AK4U;«
              «Verarbeitungsvorschrift_AKUE;«
              « Verarbeitungsvorschrift_DÜ;«
              «Funktion:= Absendequittung;
              Dialogpartner:= Benutzer;
              Verarbeitungsvorschrift_AK3N«
            Coend
            Else Cobegin {Fall 4, normaler Dialog}
              « Verarbeitungsvorschrift_DÜ;«
              «Funktion:= Absendequittung;
              Dialogpartner:= Benutzer;
              Verarbeitungsvorschrift_AK3N«
            Coend
          { In jedem Fall wird die Benutzernachricht durch DÜ an einen Nachbarn
          gesendet und zusätzlich über das Absenden eine Meldung an den Benutzer
          geschickt. Im Then-Teil muß außerdem noch ein vorangegangener
          Dialogschritt abgebrochen werden.}
        Arbeitsplatzrechner, Schullechner:
          If [Unterbrechung] in Funktion
            Then Cobegin {Fall 8 und Fall 10}
              Verarbeitungsvorschrift_AK4U;
              Verarbeitungsvorschrift_AKUE
            Coend
            Else Verarbeitungsvorschrift_AK4N {Fall3, Fall 5, normaler Dialog}
          End
        End
  End

```

Bezüglich der Unterbrechung sind die drei möglichen Nachfolgeinstanzen AK3U, AK4U und AKUE. Konnten die Instanzen AKAU, AKGU, AKWU, AKÜU nicht schalten, weil der zu unterbrechende Dialogschritt noch in Bearbeitung durch einen der Rechner war, so muß die auf mu1 gesetzte Marke abgezogen werden, um den ordnungsgemäßen Ablauf nachfolgender Dialogschritte zu gewährleisten. Dies ist Aufgabe der Instanz AKUE:

Procedure Verarbeitungsvorschrift_AKUE;

Begin

Übergabemarke_mu1 := 'undef'

End { Da in der Konfliktlösungsvorschrift_akü24 immer dann für AKUE entschieden wird, wenn "Unterbrechung" als Funktion definiert ist, zieht AKUE gerade dann die Marke von mu1 ab, wenn einer der Rechner zu unterbrechen ist. }

Die Instanz AK3U verändert die logische Struktur auf akz1 dahingehend, daß der aktuelle Arbeitszusammenhang als "undefiniert" gesetzt wird - weil bei der Unterbrechung keine weiteren Funktionen eingegeben wurden - und die logischen Informationseinheiten, die dem zuletzt aktuellen Arbeitszusammenhang zugeordnet waren, verändert werden. Zusätzlich wird eine neue logische Informationseinheit für die Unterbrechungsmeldung generiert.

Procedure Verarbeitungsvorschrift_AK3U;

Var Alt_Log_Inf_Ein, Neu_Log_Inf_Ein: **Set of** Logische_Informationseinheiten;

Alt_Log_Funk, Neu_Log_Funk: **Set of** Logische_Funktionen;

Procedure Verändere_Akt_Abstr_Dialogstruktur

(Benutzer: Benutzermodell;

Abstr_Dialogstruk: **Array**[1..Maxint] **of** Abstrakte_Dialogstruktur;

Var Akt_Abstr_Dialogstruktur: Abstrakte_Dialogstruktur;

{ In Abhängigkeit vom Benutzer und den generellen Möglichkeiten zur Bildung einer abstrakten Dialogstruktur wird die zu diesem Zeitpunkt optimale aktuelle abstrakte Dialogstruktur festgelegt. }

Procedure Suche_Log_Informationseinheiten

(Akt_Arb_Zusammenhang: Arbeitszusammenhang;

Log_Einheiten: **Set of** Logische_Informationseinheiten;

Var Betroffene_Log_Einheiten: **Set of** Logische_Informationseinheiten);

{ Die Prozedur bestimmt in der Menge aller logischer Informationseinheiten diejenigen, die einem bestimmten Arbeitszusammenhang zugeordnet sind. }

Procedure Verändere_Log_Informationseinheiten

(Benutzer: Benutzermodell;

Akt_Funk: **Set of** Logische_Funktionen;

Akt_Abstr_Dialogstruktur: Abstrakte_Dialogstruktur;

Typ: Arbeitszusammenhang;

System: Log_Funktionsumfang;

Inhalt: Text;

Konkr_Mögl: **Array**[1..Maxint] **of** Konkretisierungsmöglichkeiten;

Realis_Mögl: **Array**[1..Maxint] **of** Realisierungsmöglichkeiten;

Var Log_Inf_Ein_Alt, Log_Inf_Ein_Neu: **Set of**
Logische_Informationseinheiten;

Var Log_Funk_Alt, Log_Funk_Neu: **Set of** Logische_Funktionen);

{ Die Prozedur bestimmt in Abhängigkeit von den generellen Möglichkeiten

(Abstrakte, Konkrete Dialogstruktur, Realisierungsmöglichkeiten) und dem Benutzermodell zu den von der Unterbrechung betroffenen Logischen Einheiten neue Darstellungen }

```

Procedure Logische_Unterbrechungseinheit
  (Benutzer: Benutzermodell;
   Akt_Abstr_Dialogstruktur: Abstrakte_Dialogstruktur;
   Inhalt: Text;
   Konkr_Mögl: Array[1..Maxint] of Konkretisierungsmöglichkeiten;
   Realis_Mögl: Array[1..Maxint] of Realisierungsmöglichkeiten;
   Var Log_Inf_Einheiten: Set of Logische_Informationseinheiten;
   Var Log_Funk: Set of Logische_Funktionen);
  { Die Prozedur stellt die logischen Informationseinheiten für die
    Unterbrechungsmeldung zur Verfügung}

Begin {Verarbeitungsvorschrift_AK3U}
With Zustandsmarke_akz1.Log_Strukt do
  Begin
    If [Verändere_Akt_Abstr_Struktur] In Übergabemarke_akü24.Funktion
      Then Bestimme_Akt_Abstr_Dialogstruktur
        (Zustandsmarke_akz1.Benutzerwissen,
         Zustandsmarke_akz1.Abstrakt,
         Akt_Abstr_Dialogstruktur);
    Suche_Log_Informationseinheiten (Akt_Arbeitszusammenhang,
                                     Log_Einheiten,
                                     Log_Inf_Ein);
    Verändere_Log_Informationseinheiten
      ( Zustandsmarke_akz1.Benutzerwissen,
        [Unterbrechung],
        Akt_Abstr_Dialogstruktur,
        Akt_Arbeitszusammenhang,
        Zustandsmarke_akz1.Dialogwissen[Akt_Arbeitszusammenhang],
        Übergabemarke_akü24.Nachricht,
        Zustandsmarke_akz1.Konkret,
        Zustandsmarke_akz1.Rea,
        Alt_Log_Inf_Ein,
        Neu_Log_Inf_Ein,
        Alt_Log_Funk,
        Neu_Log_Funk);
    Logische_Einheiten:= Logische_Einheiten - Alt_Log_Inf_Ein;
    Logische_Einheiten:= Logische_Einheiten + Neu_Log_Inf_Ein;
    Akt_Funktion:= Akt_Funktion - Alt_Log_Funk;
    Akt_Funktion:= Akt_Funktion + Neu_Log_Funk;
    { vgl. die Bemerkungen in Procedure AK3N }
    Logische_Unterbrechungseinheit ( Zustandsmarke_akz1.Benutzerwissen,
                                     Akt_Abstr_Dialogstruktur,
                                     Übergabemarke_akü24.Nachricht,
                                     Zustandsmarke_akz1.Konkret,
                                     Zustandsmarke_akz1.Rea,
                                     Neu_Log_Inf_Ein,
                                     Neu_Log_Funk);
    Logische_Einheiten:= Logische_Einheiten + Neu_Log_Inf_Ein;
    Akt_Funktion:= Akt_Funktion + Neu_Log_Funk;
  End

```

```

    Zustandsmarke_akz1.Dialogpartner:= Übergabemarke_akü24.Dialogpartner;
    Akt_Arbeitszusammenhang:= 'undef'
  End;
  Übergabemarke_akü23.Log_Strukt:= Zustandsmarke_akz1.Log_Strukt;
  Übergabemarke_akü23.Akt_Arbeitsumgebung:=
    Übergabemarke_akü24.Akt_Arbeitsumgebung
End      {Verarbeitungsvorschrift_AK3U}

```

Nach Durchlaufen dieser Instanz wird die an den Benutzer zu leitende Information wie im normalen Dialog bearbeitet (vgl. letzten Abschnitt).

Die Instanz AK4U stellt - wie im normalen Dialog AK4N - im Unterbrechungsfall für den Arbeitsplatzrechner die Verbindung der Logischen Ebene zur Bearbeitungsebene her. Ansonsten übergibt sie die Information der Logischen Ebene zur Weiterbearbeitung durch SK1U an AV1. Außer dieser Unterscheidung muß die Instanz AK4U bei der Bearbeitung der Information trennen zwischen bloßer Unterbrechung eines vorhergehenden Schrittes und Unterbrechen eines vorhergehenden Schrittes und anschließendem Bearbeiten weiterer Funktionen. Wir beschreiben die Funktionsweise wie folgt:

Procedure Verarbeitungsvorschrift_AK4U;

```

  Type Speicher = Record
    Log_Funk: Set of Logische_Funktionen;
    Nachricht: Text
  End;

```

```

Procedure Fülle_Zwischenspeicher (Lg_Funk: Logische_Funktionen;
                                Zw_Speicher: Speicher);

```

{Durch die Prozedur wird die zunächst zwischenzuspeichernde Information in den Zwischenspeicher der Zustandsmarke akz2 eingetragen, wenn vor Bearbeitung der Information andere Schritte zur Vorbereitung ausgeführt werden müssen.}

```

Begin      {Verarbeitungsvorschrift_AK4U}

```

```

With Zustandsmarke_akz2 Do

```

```

  Begin    Zwischenspeicher.Funktion := [];
            Zwischenspeicher.Nachricht:= '';
            Dialogpartner:= Übergabemarke_akü24.Dialogpartner;

```

```

  End;

```

```

With Übergabemarke_akü24 Do

```

```

  If [Zwischenspeicher] in Funktion

```

```

    Then Begin

```

```

      Fülle_Zwischenspeicher (Zwischenspeicher,
                              Zustandsmarke_akz2.Zwischenspeicher);

```

```

      Funktion:= Funktion - [Zwischenspeicher]

```

```

    End;

```

{Im Bedarfsfall wird der Zwischenspeicher gefüllt und die zu bearbeitenden Funktionen um die "Zwischenspeicher"-Funktion reduziert.}

```

With Übergabemarke_akü25 Do

```

```

  Begin

```

```

    Dialogpartner:= Übergabemarke_akü24.Dialogpartner;

```

```

Form:= Kommando; {Wichtig ist eine Form ungleich der Unterbrechung}
If Zustandsmarke_akz2.Dialogrechner ≠ Dialogpartner
  Then Begin {Die Funktionen beziehen sich auf den Schulrechner; der
    Arbeitsplatzrechner muß aber unterbrochen werden.}
    Form:= Unterbrechung;
    Übergabemarke_akü24.Funktion:= Übergabemarke_akü24.Funktion -
      [Unterbrechung];
    Bearb_Strukt.Funktion:= [Unterbrechung];
    Bearb_Strukt.Nachricht:= 'Letzten Dialogschritt abbrechen';
    Bearb_Strukt.Akt_Arbeitszusammenhang:=
      Übergabemarke_akü24.Akt_Arbeitszusammenhang;
    Akt_Arbeitsumgebung:= Übergabemarke_akü24.Akt_Arbeitsumgebung
  End;
If Dialogpartner = Arbeitsplatzrechner
  Then Begin
    Bearb_Strukt.Funktion:= [Unterbrechung];
    Bearb_Strukt.Nachricht:= 'Letzten Dialogschritt abbrechen';
    Bearb_Strukt.Akt_Arbeitszusammenhang:=
      Übergabemarke_akü24.Akt_Arbeitszusammenhang;
    Akt_Arbeitsumgebung:= Übergabemarke_akü24.Akt_Arbeitsumgebung
  End
Else Begin
    Funktion:= Übergabemarke_akü24.Funktion;
    Nachricht:= Übergabemarke_akü24.Nachricht;
    Akt_Arbeitszusammenhang:=
      Übergabemarke_akü24.Akt_Arbeitszusammenhang;
    Akt_Arbeitsumgebung:= Übergabemarke_akü24.Akt_Arbeitsumgebung
  End
End
End {Verarbeitungsvorschrift_AK4U}

```

Gemäß der Konfliktlösungsvorschrift für akü25 (s.o.)

- ist AU Nachfolgeinstanz, falls nur ein Dialogschritt auf dem Arbeitsplatzrechner abzurechnen ist,
- ist AV1 Nachfolgeinstanz, falls der Arbeitsplatzrechner nicht Dialogpartner ist, oder
- es sind AU und AV1 Nachfolger und schalten parallel, falls der Arbeitsplatzrechner zu unterbrechen ist und der Schulrechner durch die Funktionen angesprochen ist.

Die Funktionsweise von AV1 wurde im letzten Abschnitt angesprochen, die der Instanz AU ergibt sich mit

Type Übergabemarke_au2 = (unterbrechen)

als

Procedure Verarbeitungsvorschrift_AU;

Begin

Übergabemarke_au1:= 'undef';

Übergabemarke_au2:= unterbrechen;

```

With Übergabemarke_akü14 Do
  Begin
    Bearb_Strukt.Akt_Arbeitszusammenhang:=
      Übergabemarke_akü25.Bearb_Strukt.Akt_Arbeitszusammenhang;
    Bearb_Strukt.Funktion:= [];
    Bearb_Strukt.Nachricht:= ' ';
    Dialogpartner:= Übergabemarke_akü25.Dialogpartner;
    Form:= Unterbrechung;
    Akt_Arbeitsumgebung:= Übergabemarke_akü25.Akt_Arbeitsumgebung
  End
End.

```

Das heißt, die Übergabemarke akü14 wird durch das Setzen des Parameters "Form" von der Unterbrechung in Kenntnis gesetzt; Funktionen und Nachrichten werden aber nicht übergeben. Diese werden für die weitere Arbeit von AK2 auch nicht benötigt: Sind nachfolgend Funktionen durch den Arbeitsplatzrechner zu bearbeiten, so werden diese von AK2 aus dem Zwischenspeicher des Zustandskanals akz2 geladen: Andernfalls ist die Markierung von akü14 "überflüssig", weil im Falle der "Parallel-Arbeit" von AU und AV1 der Schulrechner noch aktiv ist und als Folge davon akü14 zu einem späteren Zeitpunkt noch markiert wird, oder weil nur eine Unterbrechung durchzuführen war und der Benutzer eine Eingabeberechtigung erhalten hat. In diesen beiden Situationen muß die "störende" Marke von akü14 entfernt werden. Wir setzen dazu:

```

Procedure Verarbeitungsvorschrift_AUE;
  Begin
    Übergabemarke_akü14 := 'undef'
  End

```

und als Konfliktlösungsvorschrift für die Instanzen AK2 und AUE:

```

Procedure Konfliktlösungsvorschrift_akü14;
  Begin
    If (Übergabemarke_akü14.Form = Unterbrechung) and
      (Übergabemarke_akü14.Dialogpartner ≠ Arbeitsplatzrechner)
    Then Verarbeitungsvorschrift_AUE
    Else Verarbeitungsvorschrift_AK2
  End

```

Denn die Konstellation, daß der Arbeitsplatzrechner unterbrochen wurde und nicht auch Dialogpartner ist tritt nur auf, wenn eine andere Komponente des Systems (Schulrechner oder Benutzer) aktiv ist.

Neben dem eigentlichen Abbrechen durch AU müssen die zum Zeitpunkt des Unterbrechens vorliegenden Zwischenergebnisse gespeichert werden. Dies erfolgt durch eine der Instanzen AUN1 bis AUN4. Hier ist gerade diejenige schaltfähig, die außer auf au2 auch auf ihrem zweiten Eingangskanal eine Marke

(das Zwischenergebnis) findet. Die Funktionsweise dieser Instanzen läßt sich beschreiben durch

```

Procedure Verarbeitungsvorschrift_AUN1;
  Var Akt_Arb_Zus: Akt_Arbeitszusammenhang;
  Begin
    With Übergabemarke_az1 do
      Begin
        Akt_Arb_Zus:= Bearb_Strukt.Akt_Arbeitszusammenhang;
        Zustandsmarke_az.Anwendungsinhalt[Akt_Arb_Zus].Inhalt:= Bearb_Strukt;
        Zustandsmarke_az.Anwendungsinhalt[Akt_Arb_Zus].Partner:= Dialogpartner;
        Zustandsmarke_az.Anwendungsinhalt[Akt_Arb_Zus].Zustand:= 1;
        Zustandsmarke_az.Akt_Arbeitszusammenhang:= 'undef';
        Zustandsmarke_az.Akt_Arbeitsumgebung:= Akt_Arbeitsumgebung
      End
    End

```

Die Instanzen AUN2 bis AUN4 folgen entsprechend. Wieder brechen wir an dieser Stelle die Diskussion der Verarbeitungsvorschriften und Kanal-Markierungen ab, weil die Instanzen und Kanäle zur Durchführung von Unterbrechungen beim Schul- oder Großrechner entsprechend den obigen Beschreibungen organisiert sind.

14.5. Bemerkungen zu den Algorithmen

Prinzipiell läßt sich das in Abschnitt 13.4 Gesagte hier wiederholen: Wir haben mit den in den letzten vier Abschnitten hergeleiteten Algorithmen und Datenstrukturen eine Möglichkeit aufgezeigt, wie der allgemeine Dialograhmen beschrieben und realisiert werden kann. Unsere Vorgehensweise ist sicherlich nicht optimal, sondern nur eine denkbare Möglichkeit. Insbesondere ist zu bemerken:

- Wir haben zwar eine Pascal-Notation für die Algorithmen und Datenstrukturen gewählt, allerdings an vielen Stellen Pascal-Regeln verletzt. Um auf der Grundlage dieser Algorithmen und Strukturen Programme zur Realisierung des allgemeinen Dialograhmens zu entwickeln, werden umfangreiche Änderungen erforderlich sein.
- Die Algorithmen sind durch die zugrundeliegenden Ebenen sehr abstrakt und allgemein. Die eigentlichen inhaltlichen Schwierigkeiten - Was sind "Logische Funktionen"? Wie überführt man die "logische Struktur" in die "Layout-Struktur"? Wie bildet man "logische Informationseinheiten"? Wie organisiert, verwaltet und definiert man das "Benutzermodell"? - wurden nicht gelöst, sondern durch Definition von Prozedur-Köpfen übergangen.
- Die Trennung der "Layout-Ebene" von der "logischen Ebene" erscheint auf den ersten Blick willkürlich. Sicherlich werden Instanzen zur Bearbeitung und Verwaltung von Fenstern und für Reaktionen auf direkte Manipulationen benötigt, aber der Unterschied zwischen "Layout-Funktionen" und "logischen

Funktionen" wird nicht deutlich. Hier wäre eine andere Vorgehensweise möglich gewesen. Wir haben die Trennung der Ebenen auf die Funktionen übertragen, um den Transformationsprozeß von Anwendungsfunktionen zu physikalischen Funktionen über alle Ebenen gleichartig beschreiben zu können.

Dennoch ist festzuhalten, daß die gefundene algorithmische Beschreibung der dynamischen Prozesse beim Dialog mit interaktiven Systemen einen Weg aufzeigt, wie ein anwendungs- und geräteunabhängiges Dialogsystem realisiert werden kann. Die Modellierung durch eine Kombination von Petri-Netzen mit Algorithmen für Instanzenvorschriften und Datenstrukturen für Kanalmarken hat sich dabei insgesamt bewährt. Doch auch hier sind einige Aspekte noch ungelöst und wurden in den obigen Beschreibungen übergangen. Als Beispiel sei genannt:

Eine Unterbrechung führt im Modell immer zu parallelem Schalten von verschiedenen Instanzen. Wird an der Stelle AK1U festgestellt, daß der Schulrechner zum Unterbrechungszeitpunkt aktiv ist, so wird eine Unterbrechung des Schulrechners eingeleitet. Bis die diesbezüglichen Informationen den Schulrechner über AK4Ü und AV1 erreicht haben, kann durch die inzwischen stattgefundenen Schaltvorgänge beim Schulrechner aber die "normale" Marke bereits wieder zum Arbeitsplatzrechner (z.B. AV2) gelangt sein. Das Modell versagt an dieser Stelle. Notwendig wäre ein sofortiges "Stoppen", bzw. eine Art Vorrangregel: "Wenn eine Unterbrechungs- markierung sich durch das Netz bewegt, darf keine weitere Markierung parallel durch das Netz geschoben werden", die ihrerseits - in den Konfliktlösungsvorschriften wurde mehrfach paralleles Schalten ausdrücklich vorgesehen - aber auch wieder Probleme mit sich bringt. Hier ist weitere Forschungsarbeit notwendig, um im Modell auch solche Fälle formal, ohne weitere Interpretationen, berücksichtigen zu können.

15. Zusammenfassung und Ausblick

Wir haben in den drei Teilen der Arbeit zunächst eine Klassifikation des Mensch-Rechner-DIALOGs unter zwei verschiedenen Gesichtspunkten entwickelt, die gefundenen Klassen dann dazu benutzt, benutzerfreundliche Konzepte für interaktive Systeme am Beispiel des CUU zu formulieren und die Konzepte dann durch Herleitung detaillierter Modelle der Realisierung nahegebracht.

Bei den Klassifikationen (vgl. Kapitel 3) konnten wir von Untersuchungen von Dehning, Essig, Maaß /Dehning78/ ausgehen, die eine abstrakte, eine konkrete und eine technische Ebene bei der Diskussion von Mensch-Rechner-DIALOGen unterschieden haben. Wir haben ihre Ergebnisse aufgegriffen, um eine Ebene - die Programmkonzeption -, die für das Dialogverhalten von interaktiven Systemen eine bedeutende Rolle spielt, erweitert und bei der technischen Ebene einige heute allgemein anerkannte Gesichtspunkte zur Realisierung von Mensch-Rechner-DIALOGen hinzugefügt. Während wir bei der Konkretisierung von Dialogen weitgehend den Bemerkungen von Dehning, Essig, Maaß folgen konnten, haben wir auf der abstrakten Ebene umfangreiche Veränderungen vorgenommen:

Ausgehend von den Fragen, welcher der beiden Dialogpartner für einen Dialogschritt die Initiative übernimmt und inwieweit der Benutzer eigenverantwortlich handeln kann, konnten wir die abstrakten Dialogformen Datenabfrage, Auswahlangebot, Kommando und Unterbrechung als Grundelemente eines jeden Mensch-Rechner-DIALOGs angeben. Wir sind damit einen Schritt über die bisherige Betrachtungsweise für Mensch-Rechner-DIALOGe, die allein das Kriterium der Initiative berücksichtigte, hinausgegangen, wodurch der Begriff Benutzerfreundlichkeit von interaktiven Systemen um einen wesentlichen Aspekt bereichert wurde.

Die Anwendung und Umsetzung der Klassifikationen des ersten Teils im Sinne von Forderungen für Mensch-Rechner-DIALOGe war dann Inhalt des zweiten Teils der Arbeit. Hier haben wir als Ausgangspunkt den vom Dialogverhalten her bisher vernachlässigten Bereich CUU gewählt und mit Hilfe der Dialogklassifikationen eine Reihe von Konzepten zum Entwurf benutzerfreundlicher CUU-Systeme formuliert. Viele dort genannte Punkte sind in anderem Zusammenhang heute allseits anerkannt. Die Vorgehensweise, Forderungen zur Benutzerfreundlichkeit auf die Ebenen der Klassifikationen zu beziehen und durch sie zu begründen, ist jedoch neu und hat sich insofern bewährt, als in den bisher so verschwommenen Begriff der Benutzerfreundlichkeit eine "gewisse Ordnung" gebracht werden konnte. Es bleibt zu hoffen, daß für den Bereich CUU einige der genannten Konzepte aufgenommen und bei Implementierungen berücksichtigt werden.

Ein genanntes Konzept ist heute eines der Haupt-Forschungsgebiete innerhalb der Software-Ergonomie: Unter dem Namen "User-Agent" oder "Dialog-Socket" oder - wie in dieser Arbeit - "Allgemeiner Dialograhmen" werden Systeme gefordert und konzipiert, die benutzer- und anwendungsunabhängig den Dialog zwischen einem Benutzer und einem Anwendungsprogramm verwalten, Bedürfnisse der Partner erkennen und sich diesen anpassen können. Allerdings wird dabei allgemein ein Benutzer eines Computers zur Grundlage genommen. Der von uns geforderte Allgemeine Dialograhmen sieht eine Rechner-Hierarchie mit mehreren möglichen Rechnern als Partner für den Benutzer vor und geht darum wesentlich über die bisher veröffentlichten Konzepte hinaus. Entwickelt aus den Anwendungswünschen in Unterrichtsumgebungen ist solch eine Hierarchie auch allgemein auf viele Anwendungsbereiche (z.B. CAD oder Bürosysteme) als wünschenswert und notwendig übertragbar.

Der dritte Teil der Arbeit befaßt sich deswegen ausschließlich mit den Realisierungsmöglichkeiten des Allgemeinen Dialograhmens. Ausgehend von Arbeiten von Oberquelle /Oberquelle76/ formulierten wir attributierte, markierte Petri-Netze zur Modellierung des statischen und dynamischen Verhaltens von Mensch-Rechner-Dialogen und benutzten pascal-artige Algorithmen zur Beschreibung der Funktionen der Netze. Auch diese Vorgehensweise hat sich insgesamt bewährt. Wenn auch einige Probleme offengeblieben sind (vgl. Abschnitt 14.5), so konnten wir eine Darstellung angeben, die erstmals über die rein konzeptionelle Ebene für ein geräte- und anwendungsunabhängiges Dialogsystem hinausgeht und Einzelfunktionen präzisiert, ihre Bedeutung innerhalb des gesamten Systems deutlich macht und als Grundlage für Implementierungen dienen kann. Es ist allerdings an dieser Stelle noch einmal zu betonen, daß viele Probleme im Zusammenhang mit der Realisierung im Modell überdeckt wurden und hier noch vielfältige Forschungsaktivitäten bis zu einer Realisierung notwendig sind. Weiter soll hier auch nicht verschwiegen werden, daß schon das Modell zeigt, daß im Falle einer Implementierung ein sehr umfangreiches System entstehen wird, das zumindest für absehbare Zeit im Bereich CUU, aus dem heraus seine Notwendigkeit begründet wurde, wegen der für das allgemeine öffentliche Schulwesen zu hohen Kosten für eine entsprechende Hardware nicht einsetzbar sein wird.

Literaturverzeichnis

- /Artl 78/ W. Artl: Zum Lernen und Lehren mit DV-Anlagen.
In: W. Artl (Hrsg.): EDV-Einsatz in Schule und Ausbildung.
München, Wien: Oldenbourg 1978
- /Balzert86/ H. Balzert: Software Ergonomie und Software Engineering.
(Erscheint demnächst)
- /Bammé 83/ Arno Bammé u. a.: Maschinen-Menschen, Mensch-Maschinen.
Reinbek: Rowohlt Taschenbuch, 1983
- /Bauer 85/ J. Bauer, T. Schwab: Aktive Hilfesysteme.
In: H.J. Bullinger: Software Ergonomie 85. Stuttgart:
Teubner 1985
- /Bense 85/ Hermann Bense: Integration von Anwendungssystemen bei
Arbeitsplatzrechnern
(unveröffentlichtes Manuskript)
- /Benz 81/ G. Benz, R. Grob, P. Haubner: Gestaltung von
Bildschirmarbeitsplätzen. Köln: TÜV Rheinland 1981
- /Bewley 83/ W. J. Bewley, u. a.: Human Factors Testing in the Design of
Xerox's 8010 "STAR" Office Workstation.
In: Ann Janda (Ed.): Human Factors in Computing Systems.
CHI 83 Conference Proceedings. ACM, Special issue of the
Sigchi Bulletin.
- /Birster 84/ Rolf Birster: Unterrichtsmittel: Computer.
In: Journal Neue Technologien. EDV. Sonderthemenbeilage als
Supplement in: Grundschule 10/84.
- /Böcker 82/ H.D. Böcker: Intelligenter CUU und Lernumgebungen - Wege
des Rechnerunterstützten Lernens in die Zukunft?
Log In 2 (1982), 31-33
- /Brown 84/ I. C. Brown: Microcomputers in UK Primary Schools with
special reference to the concept Keyboard.
Paper presented to the Eights International Conference on
Modern Trends in Education, Plovdiv, Bulgaria, 8. - 12.
October 1984
- /Bullinger 83. H.-J. Bullinger, R. Gunzenhäuser, R. Lauber u.a.:
Rechnerunterstützte Arbeitsplätze. Antrag an die Deutsche
Forschungsgemeinschaft, Univ. Stuttgart, 1983
- /Cakir 83/ Ahmet Cakir: Über den Stand der Software Ergonomie.
Humane Produktion 12/83

- /CHI 83/ Human Factors in Computing Systems. CHI'83 Conference Proceedings, December 12.-15., Boston. Edited by Ann Janda ACM, Special issue of the Sigchi Bulletin
- /Chip 7/84/ Netzwerk fürs Computer-Training.
In: Chip, Juli 84, Nr. 7, S. 74 - 75.
- /Chip 10/84/ Komfortable Techniken.
In: Chip, Oktober 84, Nr. 10, S. 14 - 17
- /Christie85/ B. Christie (Ed.): Human Factors of the user-system interface. Amsterdam, New York, Oxford: North Holland, 1985
- /Control Data 83/ Leichter lernen. Schneller begreifen. Besser behalten. PLATO. Die beste Methode, Wissen individuell zu vermitteln. Control Data, Mexikoring 23, 2000 Hamburg 60. CBE - 83/3
- /Coutaz 85/ J. Coutaz: Abstractions for User Interface Design.
In IEEE Computer vol 5 Nr. 9, Sept. 85, 21 - 34
- /Dehning78/ W. Dehning, H. Essig, S. Maaß: Zur Anpassung virtueller Mensch-Rechner-Schnittstellen an Benutzererfordernisse im Dialog. Mitteilungen des Fachbereiches Informatik der Universität Hamburg. IFI-HH-B-50/78, 1978
- /Diebold86/ Diebold Management Report, Nr. 4 - 1986
- /DIN 66234/ DIN 66234, Teil 8: Bildschirmarbeitsplätze. Grundsätze der Dialoggestaltung. Entwurf Dezember 1984
- /Essig 81/ H. Essig: Benutzerfreundlichkeit.
Informatik Spektrum, Band 4, Heft 1, Febr. 1981
- /Fährnich 85/ K. P. Fährnich, J. Ziegler: Direkte Manipulation als Interaktionsforum an Arbeitsplatzrechnern.
In: H. J. Bullinger (Hrsg.): Software Ergonomie 85, Stuttgart: Teubner 1985
- /Foley 84/ J. O. Foley, V. L. Wallace, P. Chan: The Human Factors of Computer Graphics Interaction Techniques
IEEE Computer Graphics and Application
VOL 4, No. 11, 1984, 13 - 48
- /GKS83/ DIN 66252 Informationsverarbeitung. Graphisches Kernsystem (GKS) Funktionale Beschreibung. Entwurf Oktober 1983
- /Godbersen86/ H.P. Godbersen, H. Trümner: Ein graphisch orientierter Ansatz zur Bildung operationaler Netze.
Eingereicht als Beitrag zur GI-Jahrestagung 1986 in Berlin
- /Gorny80/ P. Gorny: Einige didaktikorientierte Bemerkungen über die Konstruktion von Dialogsystemen für naive Benutzer.
In: Notizen zum interaktiven Programmieren, Heft 4, 1980

- /Gorny 84/ P. Gorny, A. Viereck: Interaktive grafische Datenverarbeitung
Stuttgart: Teubner, 1984
- /Hansen 84/ H. R. Hansen: Mikrocomputer in der US-amerikanischen
Hochschulausbildung. Angewandte Informatik, Heft 11,
Nov. 84, 441 - 451
- /Hentig 84/ Hartmut von Hentig: Das allmähliche Verschwinden der
Wirklichkeit. München, Wien: Hanser 1984
- /Hoffmann 79/ Gerd Hoffmann: Computer, Macht und Menschenwürde.
Frankfurt: Fischer Taschenbuch, 1979
- /Horlacher82/ E. Horlacher: Die Verteilung von Programmen zum
rechnergestützten Lernen über öffentliche Kommuni-
kationsmedien.
In: J. Nehmer(Hrsg.): GI-12. Jahrestagung. Berlin,
Heidelberg, New York: Springer 1982
- /Hubert 83/ Eva Hubert: Politiker fragen - Bürger antworten nicht!
In: J. Taeger (Hrsg.): Die Volkszählung.
Reinbek bei Hamburg: Rowohlt, 1983
- /Intus 84/ Lernprogramme für die Schule für Apple II Computer.
Katalog 1984. INTUS Lern-Systeme AG,
Bellevueweg 5, CH-6300 Zug/Switzerland
- /Kaier 77/ E. Kaier: Lehren, Lernen und Computer.
Stuttgart: Deutsche Verlags Anstalt, 1977
- /Keil 78/ K.-A. Keil: CUU-Einsatz in der Schule - Leistungen und
Empfehlungen
In: W. Artl (Hrsg.): EDV - Einsatz in Schule und Ausbildung
- /Klein 85/ H.J. Klein, K. Schertel: Designprinzipien der Gestaltung
von Benutzer-System-Schnittstellen.
In: Mandl/Fischer (Hrsg): Lernen im Dialog mit dem
Computer. München, Wien, Baltimore: Urban &
Schwarzenberg, 1985
- /Klett 83/ Schulsoftware - Katalog apple. Verzeichnis privater und
kommerzieller Anwendungsprogramme für den Schulbereich.
Stuttgart: Klett, 1983
- /Kühn80/ M. Kühn: CAD und Arbeitssituation.
Berlin, Heidelberg, New York: Springer 1980
- /Kupka 81/ J. Kupka, S. Maaß, H. Oberquelle: Kommunikation - ein
Grundbegriff für die Informatik. Mitteilungen des FB
Informatik der Universität Hamburg
IFI-HH-M-91/81, August 1981

- /Kupka 82/ J. Kupka, S. Maaß, H. Oberquelle: Kommunikation in Mensch-Rechner-Dialogen.
In: J. Nehmer (Hrsg.): GI - 12. Jahrestagung.
Berlin, Heidelberg, New York: Springer 1982
- /Lenk 82/ Klaus Lenk: Informationstechnik und Gesellschaft.
In: G. Friedrichs, A. Schaff (Hrsg.): Auf Gedeih und Verderb. Mikroelektronik und Gesellschaft. Bericht an den Club of Rome. Wien: Europa Verlag, 1982
- /Lutze 85/ R. Lutze: Hilfesysteme - Ihre Beziehung zu Anwendungssystemen und zukünftige Entwicklungstendenzen.
In: H. J. Bullinger (Hrsg.): Software Ergonomie 85, Stuttgart: Teubner, 1985
- /Maaß 84/ S. Maaß: Mensch-Rechner-Kommunikation-Herkunft und Chancen eines neuen Paradigmas. Universität Hamburg, FB Informatik, IFI-HH-B.104/84, Juli 1984
- /Marcus 84/ A. Marcus: Corporate Identity for Iconic Interface Design: The Graphic Design Perspective. IEEE Computer Graphics and Application VOL 4, No. 12, (Dec. 84), 24 - 32
- /Myers 84/ Brad A. Myers: The User Interface for Sapphire. IEEE Computer Graphics and Application Vol 4, No. 12, 13 - 23
- /Nievergelt 83/ J. Nievergelt, A. Ventura: Die Gestaltung interaktiver Programme. Stuttgart: Teubner, 1983
- /Oberquelle 78/ H. Oberquelle: "Grobe" Beschreibung von Systemen durch Netze. Mitteilungen des FB Informatik der Universität Hamburg. IFI-HH-M-62/78
- /Oberquelle 76/ H. Oberquelle: Grundbegriffe zur Beschreibung von Dialogen und dialogfähigen Systemen. Bericht Nr. 28, IFI-HH-B-28/76 des Instituts für Informatik der Universität Hamburg, 1976
- /Rank Xerox85/ Xerox-Netzwerk-System. Rank-Xerox GmbH, 1985
- /Reisig 82/ W. Reisig: Petrinetze. Eine Einführung:
Berlin, Heidelberg, New York: Springer 1982
- /Saathoff 82/ Annette Saathoff: Formale Modelle für den Mensch-Rechner-Dialog. Diplomarbeit 1982 im FB Mathematik der Universität Oldenburg
- /Schäfer86/ G. Schäfer: MEDIE: Method for Dialog Engineering. Eingereicht als Beitrag zur GI-Jahrestagung 1986 in Berlin.
- /Schmitt 83/ A. Schmitt: Dialogsysteme. BI-Wissenschaftsverlag Mannheim, Wien, Zürich 1983

- /Schneider 83/ Schneider, H. J. (Hrsg.): Lexikon der Informatik und Datenverarbeitung. München, Wien: Oldenbourg, 1983
- /Schneider85/ M.Schneider, M.Herczeg, H. Wenzel: Designkriterien für benutzermodifizierbare Systeme.
In: Computer Magazin 10/85, 46-55
- /Schulze 84/ H. H. Schulze: Das RORORO Computer Lexikon. Reinbeck bei Hamburg: Rowohlt, 1984
- /Schackel 85/ B. Schackel: Human Factors and Usability - Whence and Whither.
In: H. J. Bullinger (Hrsg.): Software Ergonomie 85, Stuttgart: Teubner, 1985
- /Simitis 84/ S. Simitis: Reicht unser Datenschutz angesichts der technischen Revolution? - Strategien zur Wahrung der Freiheitsrechte.
In: Informationsgesellschaft oder Überwachungsstaat. Protokoll zum Symposium der Hessischen Landesregierung. 3. - 5. September 1984. Herausgegeben vom Hessendienst der Staatskanzlei, Postfach 31 47, 6200 Wiesbaden.
- /Spiegel 47/84/ Alarm in den Schulen: Die Computer kommen.
Titelgeschichte. Der Spiegel, Nr. 47, 1984 Nov. 84, S. 97 - 129
- Staufer 85/ M. J. Staufer: Piktogramme für Bürocomputer.
Forschungsbericht Verbundprojekt WISDOM.
Triumph-Adler AG Basisentwicklung Juni 1985
- /Stelovsky 85/ J. Stelovsky, H. Sugayn: Command Language vs. Menus or Both?
In: H. J. Bullinger (Hrsg.): Software Ergonomie85, Stuttgart: Teubner, 1985
- /Stork 85/ Burkhard Stork: Services als funktional abstrakter Kern eines Softwarearbeitsplatzes.
In: Klopčic, Marty, Rothauser (Hrsg): Arbeitsplatzrechner in der Unternehmung, Stuttgart: Teubner, 1985
- /Streitz 85/ N. Streitz: Die Rolle von mentalen und konzeptuellen Modellen in der Mensch-Computer-Interaktion: Konsequenzen für die Software Ergonomie?
In: H. J. Bullinger (Hrsg.): Software Ergonomie 85, Stuttgart: Teubner, 1985
- /Streitz 85a/ N.Streitz: Kognitionspsychologische Aspekte der Gestaltung von Dialogstrukturen bei interaktiven Lehr-Lern-Systemen.
In: Mandl/Fischer (Hrsg): Lernen im Dialog mit dem Computer. München, Wien, Baltimore: Urban & Schwarzenberg, 1985
- /Swezey 83/ Swezey, R. W.; Davis, E. G.: A case study of Human Factors Guidelines in computer graphics.

- IEEE Computer Graphics and Application
vol. 3, no. 8 (Nov. 1983) 21 - 30
- /Telefunken 73/ PLANIT Unterrichts-Dialogsprache. Einführung.
Telefunken Computer GMBH, 7750 Konstanz,
Max-Strohmeyer Str. 16 Best. Nr. 490.69.02/3.73
- /van der Veer 84/ G. C. van der Veer, u. a. (Hrsg.): Readings on cognitive
Ergonomics - Mind and Computers. Berlin, Heidelberg, New
York, Tokio: Springer, 1984
- /van Weert 84/ T. van Weert: Modell-Lehrplan "Informatische Grund-
kenntnisse für alle Lehrer"
Log In 4 (1984), 36 - 43
- /Viereck 86/ A. Viereck: Klassifikation des Mensch-Rechner-DIALOGs
für eine benutzerfreundliche Dialoggestaltung.
In: Notizen zu interaktiven Systemen, Heft 15, März 1986
- /Wagner 82/ K. Wagner: Mikrorechner im Unterricht am Beispiel der
Physik.
In: J. Nehmer (Hrsg.): GI - 12. Jahrestagung. Berlin,
Heidelberg, New York: Springer, 1982
- /Wedekind 81/ J. Wedekind: Unterrichtsmedium Computersimulation.
Weil der Stadt: Lexika 1981
- /Weizenbaum 78/ Joseph Weizenbaum: Die Macht der Computer und die
Ohnmacht der Vernunft. Frankfurt: Suhrkamp 1978
- /Whiteside 85/ J. Whiteside u. a.: User performance with Command, Menu and
Iconic Interfaces. Proc. of. CHI'85: Human Factors in
Computing Systems. San Francisco, April 1985
- /Zimmermann 84/ Rolf Zimmermann: Computer im Mathematikunterricht der
Sekundarstufe 1.
In: Peschke, Hullen, Diemer (Hrsg.) Anforderungen an neue
Lerninhalte. Band 1: Ergebnis der Fachtagung "Mikro-
elektronik und Schule III", Wiesbaden, 27. und 28. März 1984
Best. Nr. 50 174 Hessisches Institut für Bildungsplanung und
Schulentwicklung (HIBS), Wiesbaden 1984

Bildungsgang

Ich bin am 8. Oktober 1952 in Bremen geboren, wo ich in der Zeit von 1959 bis 1972 auch meine Schulausbildung an der Grundschule und am Gymnasium erhielt. Nach dem Abitur und einer dreijährigen Berufsausbildung begann ich 1975 mein Studium im Diplom-Studiengang Mathematik an der Universität Oldenburg. Im Rahmen des Hauptstudiums wählte ich Angewandte Informatik als Studienschwerpunkt und konnte das Mathematik-Studium mit einer Diplomarbeit über Datenbanken 1981 beenden.

Danach immatrikulierte ich mich an der Universität Oldenburg als Promotions-Student und arbeitete an verschiedenen Themen aus dem Bereich der angewandten Informatik. Zusammen mit Prof. Dr. Peter Gorny erstellte ich in dieser Zeit ein Lehrbuch zur interaktiven grafischen Datenverarbeitung.

Mit dem Aufbau des Diplom-Studienganges Informatik an der Universität Oldenburg wurde ich 1984 als wissenschaftlicher Mitarbeiter im Fachbereich Informatik der Universität Oldenburg eingestellt.

Axel Viereck

